1993

Collision-free path planning for robots using B-splines and simulated annealing

Horacio Martínez-Alfaro
Iowa State University

3

94

05057

Order Number 9405057

Collision-free path planning for robots using B-splines and simulated annealing

Martínez-Alfaro, Horacio, Ph.D.

Iowa State University, 1993

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

Collision-free path planning for robots using B-splines and simulated annealing

by

Horacio Martínez-Alfaro

A Dissertation Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Department: Mechanical Engineering
Major: Mechanical Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1993

DEDICATION

This dissertation is dedicated to my wife, Graciela, for her endless trust, encouragement, support and love during my graduate studies at Iowa State; and to the memory of my parents, Miguel and Zenaida.

Esta tesis la dedico a mi esposa, Graciela, por su eterna confianza, ánimo, apoyo y amor durante mis estudios doctorales en Iowa State University; y a la memoria de mis padres, Miguel y Zenaida.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

Thanks to Consejo Nacional de Ciencia y Tecnología (CONACYT), Institute of International Education (Fullbright-IIE), and Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM) for granting me a fellowship to pursue my graduate studies at Iowa State University.

I thank Dr. Donald R. Flugrad for his trusting support and guidance while I was pursuing this research work. I will always be deeply indebted to him.

I would also like to thank Professors James E. Bernard, Martin J. Vanderploeg, Bion L. Pierson, Vijay Vittal, and Kenneth G. McConnell for serving on my committee and for everything they taught me.

Many thanks to Dr. James Oliver for helping me with the initial setup of the computer program.

Finally, I thank my brothers and their families: Miguel, Rosy, and Edna, and Sergio, Graciela, and Sergio; as well as my sister, Graciela, for their blessings, love, encouragement and support for many years. Without these, I would be lost.

Finalmente, les agradezco a mis hermanos y familias: Miguel, Rosy y Edna y Sergio, Graciela y Sergio; asi como a mi hermana Graciela por sus bendiciones, cariño, ánimo y apoyo durante muchos años. Sin ésto, no se que sería de mi.

1. INTRODUCTION

Many manufacturing processes accomplished with robots such as welding, painting, machine loading, assembly, etc., as well as mobile robots require a smoothly varying curve. This curve may be used, for example, to represent the path of a point moving through space, as in robotic path planning tasks. Most of the techniques used to accomplish this are characterized by emphasizing the global geometric environment within which the curve must perform, or local intrinsic shape properties of the curve [11, 88, 152]. Sequences of points connected linearly (or with arcs) are used to construct the desired path [13, 14, 56, 63, 68, 80, 165]. One problem in these methods is that the curvature of the path is usually discontinuous at transition points of segments. For most robots or autonomously guided vehicles, these instantaneous changes will make continuous smooth motions physically impossible. That is, they do not provide the smooth variation, continuous derivatives, local properties, and other advantages of parametric curves.

A solution for this problem is to use a continuous-curvature curve. A technique shown in [66] uses a "clothoid pair" to make curves with zero curvature at their junctions. Another type of continuous-curvature curve is the "cubic spiral" curve proposed in [67] for being relatively simple to generate. In addition, a method called "potential field" [59, 71, 115, 175] is also developed. In this method, positive potential

fields are placed around obstacles and negative (attractive) potential field at the goal position. The sum of the resulting field is used to choose an appropriate path. Another solution to this is proposed in [186] which uses B-splines to represent the curve. A better methodology is proposed in [105] which uses also B-splines and simulated annealing to obtain an optimal path.

It is worth mention that, since a point is assumed as the moving object, those techniques do not consider its orientation. In addition, most of the work to date in path planning is just in two dimensions [14, 17, 25, 34, 43, 52, 59, 63, 98–101].

In this work, we will simultaneously address independent goals of global obstacle avoidance and local control of intrinsic shape properties by formulating a simulated annealing (SA) problem for curve synthesis. In its original form, simulated annealing is a probabilistic combinatorial optimization technique based on an analogy to the statistical mechanics of disordered systems [76]. In the physical process of annealing, as explained in [86], a material is heated and allowed to cool slowly by decreasing the temperature, so that it reaches thermal equilibrium at each of those temperatures. As a consequence, its atoms will reach a minimum energy state, the ground state, despite any local minima. In simulated annealing, the objective function or cost function to be minimized is analogous to the total energy of the system, and the values of the independent variables determine the state of the system. The variables are randomly perturbed, and if a lower cost is obtained, the new state is accepted. If a higher cost is generated, the new state is accepted with a probability of the current temperature. The SA algorithm mimics the physical process of annealing since the probability of accepting a higher cost decreases with temperature. The behavior of the SA has been characterized as first following the gross behavior of the objective

function to find an area in its domain where a global minimum should be present, irrespective of the local minima found on the way [86]. It progressively develops finer details, finding a good, near-optimal local minimum if not the global minimum itself.

One of the main characteristics of this work is that objects are modeled as ellipsoid type shapes in the work space. There exists a previous work by Chen and Vidyasagar (1988) in which objects were modeled as ellipsoids too but in the joint space of the manipulator. The techniques presented in many research works (including [14, 34, 63, 98, 99, 105, 140, 186]) assume a *point* as an object. This work goes further by considering an *actual* object to synthesize the path in two and three dimensions. The independent variables are the coordinates of B-spline control points. Three possible orientations for the moving object are analyzed: a) fixed orientation, b) orientation as another independent variable, and c) orientation given by first derivative of the curve. A proximity cost function, a component of the objective function, is developed to determine object proximity or interference with fixed objects (obstacles). Kinematic characteristics of a fixed manipulator can be incorporated to this function in order to solve the path planning problem for an object. A two-link planar manipulator is discussed to show this feature.

This thesis is organized into 7 additional chapters. Chapter 2 presents a detailed procedure to obtain the parameters that define the ellipsoids representing objects. Chapter 3 reviews material on B-splines. Chapter 4 describes the procedure to synthesize the B-spline curve in which the cost function is included. Chapter 5 describes the procedure used to find the distance and/or interference between ellipsoid shaped objects. Chapter 6 gives a description of the simulated annealing algorithm and its implementation. Chapter 7 discusses some application examples for each orientation

case in two and three dimensions to demonstrate the optimality of the algorithm. Chapter 8 presents conclusions and suggestions for further research.

2. ELLIPSOID MODEL

2.1 Ellipsoid Parameter Models

This work is based on describing objects by ellipsoid type shapes. The ellipsoid type shape was chosen because of its simplicity and its better use of space (compared to a circle). Each object, including manipulator links, is surrounded by an ellipsoid of minimum area (2D) or minimum volume (3D). These approximations are based on minimizing an easy-to-understand objective function. However, an alternate procedure to find the parameters of the ellipsoid is also described. These parameters are computed by a different module and are provided as input data to our main path planning program. The procedures are described in Section 2.2. For now, we will describe the shape first in 2D and then in 3D.

2.1.1 Ellipsoid Shape in 2D

The parameters used to describe an ellipsoid in 2D, i.e. ellipse, are (see Figure 2.1):

- Principal semi-axis dimensions: a for major semi-axis, and b for minor semi-axis.
- Orientation angle (counterclockwise direction from major semi-axis): θ.

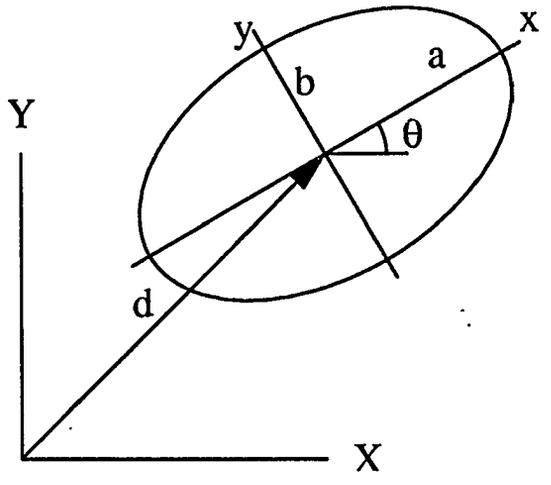


Figure 2.1: Ellipse parameter definition

- Translation vector: $\mathbf{d} = [d_x,\ d_y]^T$

The principal semi-axis dimensions define the size of the ellipse which is given by the following equation

$$f_e(x, y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0 \tag{2.1}$$

This is an ellipse with its origin located at $(0, 0)$ in $\Re^2$. The orientation angle, θ, and translation vector, $\mathbf{d}$, define the orientation and position of the ellipse relative to a fixed frame. Suppose $\mathbf{r} = [x,\ y]^T$; then, the new position of $\mathbf{r}$, i.e. $\hat{\mathbf{r}} = [\hat{x},\ \hat{y}]$, is given by

$$\hat{\mathbf{r}} = \mathbf{A}\mathbf{r} + \mathbf{d}, \quad \text{where} \quad \mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2.2}$$

Note that $\mathbf{A}$ is an *orthonormal* matrix with the property $\mathbf{A}^{-1} = \mathbf{A}^T$.

2.1.2 Ellipsoid Shape in 3D

The parameters used to describe an ellipsoid in 3D are:

- Principal semi-axis dimensions: a for x semi-axis, b for y semi-axis, and c for z semi-axis, $a > b$, $a > c$.
- Angular orientation: Euler angles ψ, θ, and ϕ with the x *convention* [114] as rotation sequence (described bellow).
- Translation vector: $\mathbf{d} = [d_x,\ d_y,\ d_z]^T$

These parameters are computed by a different module and are provided as input data to our main path planning program. The procedure is describe in Section 2.2. The principal semi-axis dimensions define the size of the ellipsoid which is given by the following equation

$$f_e(x, y, z) = \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0 \tag{2.3}$$

For the angular orientation, Figure 2.2(d) shows the initial and final systems of xyz axes. The sequence of rotation employed in the x convention starts by rotating the initial system of xyz axes counterclockwise about z axis by an angle ψ, as shown in Figure 2.2(a), and the resulting coordinate system is labeled $x'y'z'$. In the second step the intermediate $x'y'z'$ axes are rotated about x' counterclockwise by an angle θ to produce another intermediate set, the $x''y''z''$ axes (Figure 2.2(b)). Finally, the $x''y''z''$ axes are rotated counterclockwise about z'' by an angle ϕ to produce a desired XYZ system of axes (Figure 2.2(c)). The Euler angles completely specify the orientation of the XYZ system relative to the xyz system (Figure 2.2(d)). Figure 2.3 shows an ellipsoid following the sequence of rotations in the x convention.

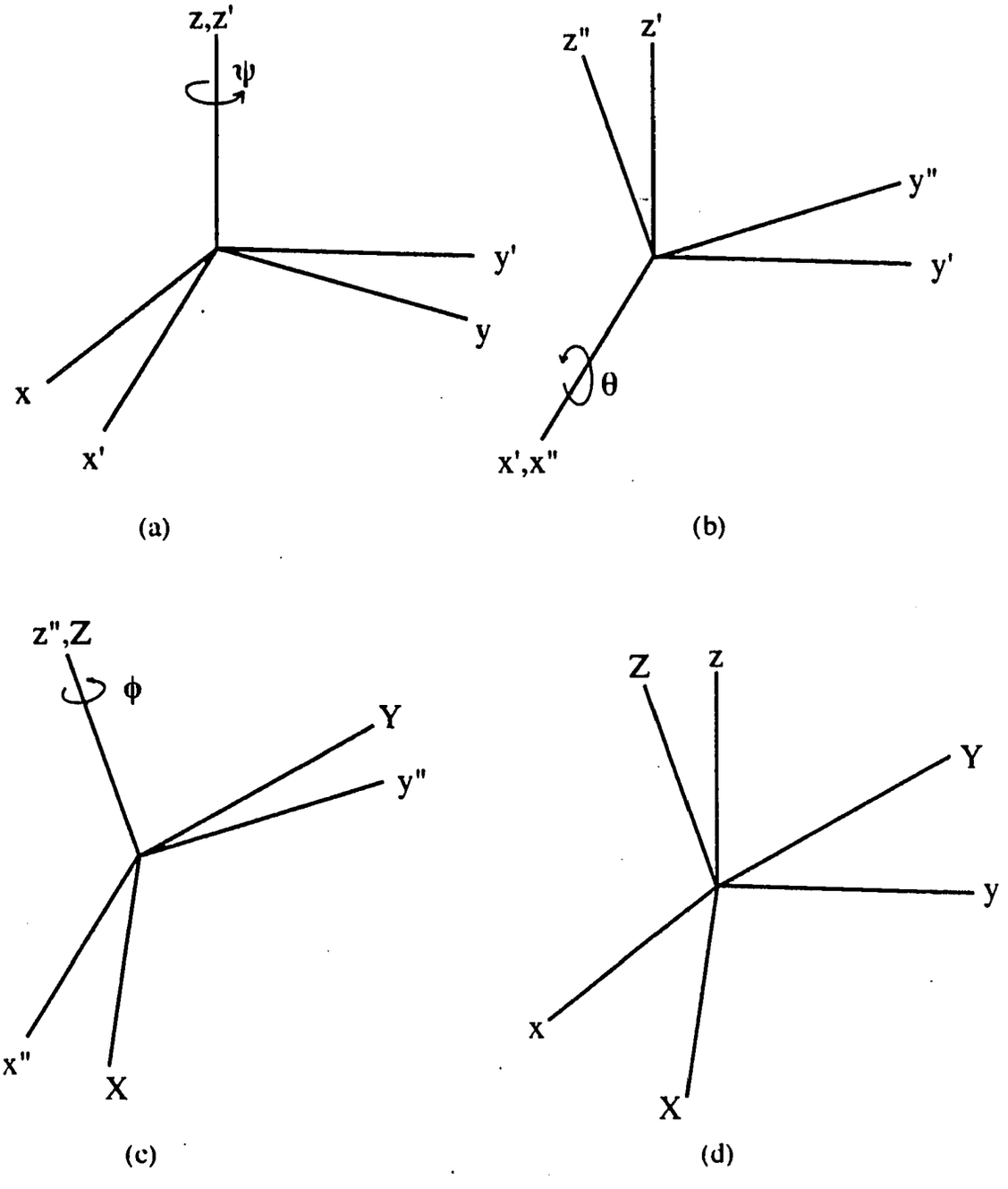
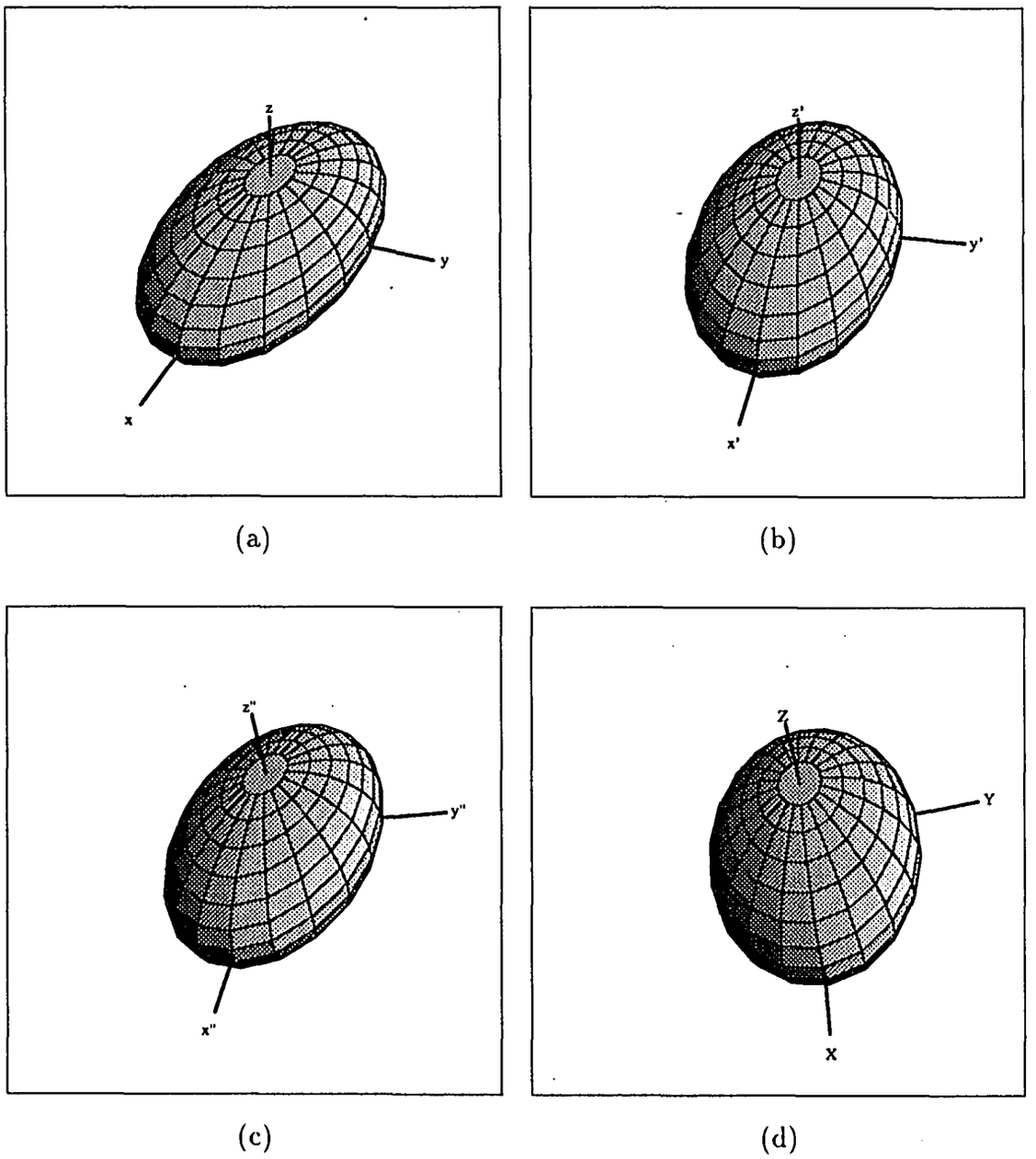


Figure 2.2: The rotations defining the Euler Angles



(a) (b)

(c) (d)

Figure 2.3: An ellipsoid following the sequence of rotations in the x convention

The elements of the complete transformation matrix **A** can be obtained as the triple product of the matrices that define the separate rotations, i.e., the matrices

$$\mathbf{D} = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\theta & -s_\theta \\ 0 & s_\theta & c_\theta \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

where $c \equiv \cos$ and $s \equiv \sin$. Hence, $\mathbf{A} = \mathbf{DCB}$ is found to be

$$\mathbf{A} = \begin{bmatrix} c_\psi c_\phi - s_\psi c_\theta s_\phi & -c_\psi s_\phi - s_\psi c_\theta c_\phi & s_\psi s_\theta \\ s_\psi c_\phi + c_\psi c_\theta s_\phi & -s_\psi s_\phi + c_\psi c_\theta c_\phi & -c_\psi s_\theta \\ s_\theta s_\phi & s_\theta c_\phi & c_\theta \end{bmatrix} \tag{2.5}$$

It can be verified that matrix **A** is orthonormal, i.e., that $\mathbf{A}^{-1} = \mathbf{A}^T$.

The transformed and translated ellipsoid follows the same relationship shown in Equation 2.2 with $\mathbf{r}$, $\hat{\mathbf{r}}$, $\mathbf{d} \in \Re^3$ and **A** given by Equation 2.5.

2.2 Procedure to Obtain Ellipsoid Parameters

It is clear from Section 2.1.1 and 2.1.2 that in order to approximate a surrounding ellipsoid, we need to find the best parameters that give a minimum area or volume. We will describe this procedure for 2D and then for 3D. Finally, an optional procedure to find the parameters is also discussed.

2.2.1 Procedure for 2D

We need to find the parameter vector

$$\tau \equiv [a,\ b,\ \hat{d}_x,\ \hat{d}_y,\ \theta]^T \tag{2.6}$$

that gives an ellipse with minimum area, $A_e(a,b)$, enclosing the object. The area was chosen as the objective function because it is easy to compute, especially for polygons. In order to accomplish this approximation, we, first *sample* the perimeter of the object. That is, we measure the coordinates of some number of points, N_s, located on the perimeter of the object with respect to any xy-coordinate system. This coordinate system could be located anywhere in $\Re^2$; however, an easy way of locating it is at one of the vertices of the object if it has one.

Since the samples are in the $\hat{x}\hat{y}$-coordinate system, we express them in the xy-coordinate system. Equation 2.2 establishes a relationship between the xy and the $\hat{x}\hat{y}$-coordinate systems. Therefore, we can solve for $\mathbf{r}$ (which is in the xy-coordinate system). Thus,

$$\mathbf{r} = \mathbf{A}^{-1}\hat{\mathbf{r}} - \mathbf{A}^{-1}\mathbf{d} \tag{2.7a}$$

$$= \hat{\mathbf{A}}\hat{\mathbf{r}} - \hat{\mathbf{d}} \tag{2.7b}$$

where

$$\hat{\mathbf{A}} = \mathbf{A}^{-1} \quad \text{and} \quad \hat{\mathbf{d}} = \mathbf{A}^{-1}\mathbf{d}$$

In order to ensure that the object is enclosed by the approximating ellipse, the condition

$$h_i(\tau, \mathbf{r}_i) = (bx_i)^2 + (ay_i)^2 - (ab)^2 \leq 0, \quad i \in \{1, \ldots, N_s\} \tag{2.8}$$

should be imposed. Thus, the problem of finding the *best* approximating ellipse can

be formulated as the following optimization problem:

$$\begin{aligned} &\text{Problem} && \min \quad A_e(a,b) - A_o \\ & && \tau \in \Re^5 \\ &\text{subject to the constraints} \\ & && h_i(\tau, \mathbf{r}_i) \leq 0, \quad i = 1, \ldots, N_s \end{aligned} \tag{2.9}$$

where $A_e(a,b) = \pi ab$ is the area of the ellipse and A_o is the area of the object.

In solving problem (2.9) numerically, the gradient vector $\partial h_i(\tau, \mathbf{r}_i)/\partial \tau$ is often required. The derivative of h_i with respect to θ is quite cumbersome to compute because of the presence of the trigonometric functions in θ. It is possible to avoid this difficulty by treating c_θ and s_θ as part of the parameter vector, τ, instead of just θ. In this case, the vector τ should be redefined as $[a,\ b,\ \hat{d}_x,\ \hat{d}_y,\ c_\theta,\ s_\theta]^T$ with the additional constraint $c_\theta^2 + s_\theta^2 - 1 = 0$.

There are several commercial routines for solving the above problem. The one used here was routine E04UCF from the NAG (Numerical Algorithm Group) library. This routine uses a sequential quadratic programming (SQP) algorithm in which the search direction is the solution of a quadratic programming (QP) problem. The algorithm treats bounds, linear constraints and nonlinear constraints separately. The user must provide routines that define the objective and constraint functions and as many of their first partial derivatives as possible. Unspecified derivatives are approximated by finite differences.

The number of samples, N_s, could be any number, although the procedure requires *at least* a minimum number of points describing the geometry of the object, for example, a square could be represented by 4 points.

2.2.2 Procedure for 3D

For the 3D model we need to find the parameter vector

$$\tau \equiv [a,\ b,\ c,\ \hat{d}_x,\ \hat{d}_y,\ \hat{d}_x,\ c_\psi,\ s_\psi,\ c_\theta,\ s_\theta,\ c_\phi,\ s_\phi]^T \tag{2.10}$$

that gives an ellipsoid with minimum volume, $V_e(a,b,c)$, enclosing the object. Here ψ, θ, and ϕ are the Euler angles that define the angular orientation of the ellipsoid and $c_{(\cdot)} = \cos(\cdot)$ and $s_{(\cdot)} = \sin(\cdot)$. The procedure to find the approximating ellipsoid is quite similar to the one described in Section 2.2.1 but now each sample has x, y, and z components, i.e., $\hat{\mathbf{r}} \in \Re^3$.

In order to ensure that the object is enclosed by the approximating ellipsoid, the conditions

$$h_i(\tau, \mathbf{r}_i) = (bcx_i)^2 + (acy_i)^2 + (abz_i)^2 - (abc)^2 \leq 0, \quad i \in \{1, \ldots, N_s\} \tag{2.11}$$

and

$$c_\psi^2 + s_\psi^2 - 1 = 0 \tag{2.12}$$

$$c_\theta^2 + s_\theta^2 - 1 = 0 \tag{2.13}$$

$$c_\phi^2 + s_\phi^2 - 1 = 0 \tag{2.14}$$

should be imposed. Thus, the problem of finding the *best* approximating ellipsoid can be formulated as the following optimization problem:

$$\begin{aligned} &\text{Problem} && \min \quad V_e(a,b,c) - V_o \\ &&& \tau \in \Re^{12} \\ &\text{subject to the constraints} \\ &&& h_i(\tau, \mathbf{r}_i) \leq 0, \quad i = 1, \ldots, N_s \end{aligned} \tag{2.15}$$

where V_o is the volume of the object.

The same routine, E04UCF, described in Section 2.2.1, was used to numerically solve the above problem.

2.2.3 Optional Procedure

We still need to find the parameter vector, τ, in order to find the approximating ellipsoid. However, instead of using the area (or volume) we could choose the *error of approximation* as the objective function for our optimization problem. If the least squares criterion is used, then the error of the approximation can be defined as

$$e(\tau) = \sum_{i=1}^{N_s} \left[\left[y_i^2 - y(x_i) \right]^2 + \left[x_i^2 - x(y_i) \right]^2 \right] \tag{2.16}$$

where $x(\cdot)$, $y(\cdot)$, x_i, and y_i are obtained from Equations 2.1, 2.2, and 2.3, i.e. (2D case)

$$\begin{aligned} x_i &= c_\theta \hat{x}_i + s_\theta \hat{y}_i - \hat{d}_x, \qquad y_i = -s_\theta \hat{x}_i + c_\theta \hat{y}_i - \hat{d}_y, \\ x(y) &= a^2 \left[1 - (y/b)^2 \right], \qquad y(x) = b^2 \left[1 - (x/a)^2 \right]. \end{aligned}$$

Note that the error is defined to be the sum of the errors associated with the x and y coordinates because there is no reason to penalize one more than the other.

2.3 Examples

Two shapes are considered here to show some results of the previously described procedure to find the parameters of the ellipsoid. The procedure proved to be sufficiently robust independent of the number of samples, N_s. We start with a rectangular shaped object.

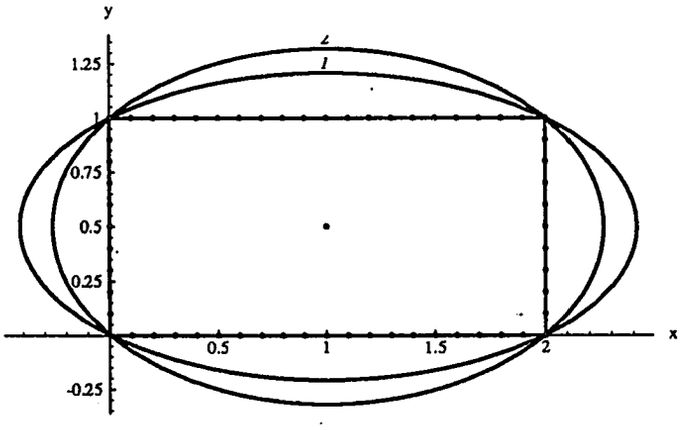


Figure 2.4: Object and approximating ellipses with $N_s = 60$

2.3.1 Example 1

Figure 2.4 shows the object and the samples. A minimum number of samples, N_s, is required by the procedure in order to be able to find the best parameters of the enclosing ellipse. Although N_s for this case is easy to determine by visual inspection, that is not always the general case. If the shape of the object is awkward, several numbers of samples should be taken and processed to determine the best approximation.

We will proceed with our example by taking samples 0.1 apart marked by dots in Figure 2.4. Since E04UCF is an iterative routine, initial values for the parameters must be chosen to start the search. Table 2.1 shows the initial values and final results of the parameters for each procedure. It also shows the number of iterations taken to reach the optimal solution. Figure 2.4 shows the object and both approximating ellipses 1 and 2 obtained with the procedures describe in Section 2.2.1 and 2.2.3, respectively. The Appendix shows output listings for each example presented in this section.

Table 2.1: Initial values and results for the parameters in Figure 2.4 with $N_s = 60$

Parameter	Initial	Results 1	Results 2
a	1.50	1.414214	1.262782
b	0.75	0.707107	0.818803
d_x	2.00	1.0	1.0
d_y	1.00	0.5	0.5
$\theta°$	0.53	0.0	0.0
A_e	3.53	3.141593	3.24831
Iter		58	19

We will continue with the same object but this time with only 4 samples of the perimeter of the object, i.e., its vertices. Figure 2.5 shows the object, the samples, and the approximating ellipses. Table 2.2 shows the initial values and the final results for the parameters.

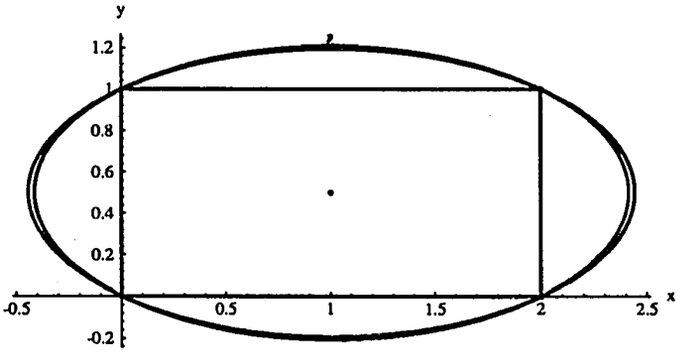


Figure 2.5: Object and approximating ellipses with $N_s = 4$

Table 2.2: Initial values and results for the parameters in Figure 2.5 with $N_s = 4$

Parameter	Initial	Results 1	Results 2
a	1.50	1.414214	1.44224
b	0.75	0.707107	0.69388
d_x	2.00	1.0	1.0
d_y	1.00	0.5	0.5
$\theta°$	0.53	0.0	0.0
A_e	3.53	3.141593	3.143922
Iter		32	25

2.3.2 Example 2

Our next example is the same figure described in Section 2.3.1 but with a certain angular orientation, i.e., it is rotated about its lower-left vertex as shown in Figure 2.6. Also shown in Figure 2.6 are the approximating ellipses marked as 1 and 2. Table 2.3 shows the initial values and final results for $N_s = 60$.

Figure 2.7 shows the approximating ellipses and Table 2.4 shows the initial values and results for the same object with $N_s = 4$. We can see in this example that the approximating ellipse (2), given by the procedure described in Section 2.2.3, is not sufficiently robust when the number of samples is small even though the shape is very simple. This characteristic could help us to conclude that the procedure described in Section 2.2.1 is better than the procedure described in Section 2.2.3.

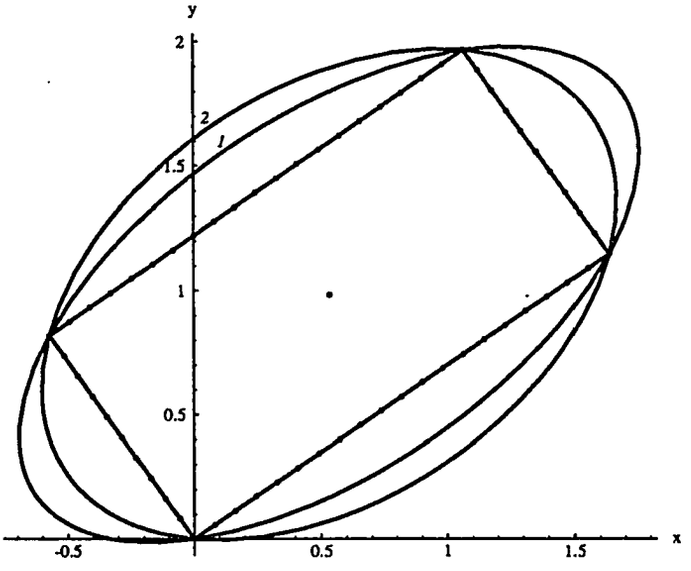


Figure 2.6: Object and approximating ellipses with $N_s = 60$ and non-zero angular orientation

Table 2.3: Initial values and results for the parameters in Figure 2.6 with $N_s = 60$ and nonzero angular orientation

Parameter	**Initial**	**Results 1**	**Results 2**
a	1.50	1.414211	1.262781
b	0.75	0.707105	0.818800
d_x	2.00	0.532361	0.532362
d_y	1.00	0.983149	0.983150
$\theta°$	0.53	35.000000	34.990000
A_e	3.53	3.141579	3.143922
Iter		16	16

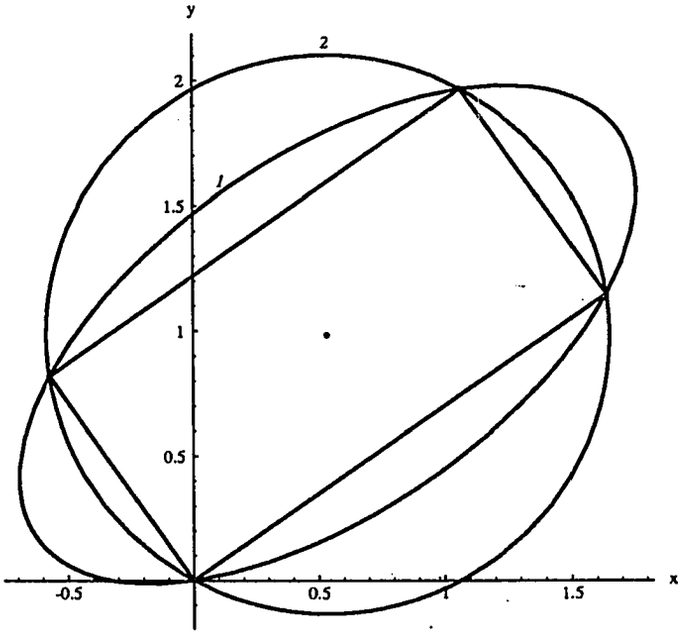


Figure 2.7: Object and approximating ellipses with $N_s = 4$ and non-zero angular orientation

Table 2.4: Initial values and results for the parameters in Figure 2.7 with $N_s = 4$ and nonzero angular orientation

Parameter	**Initial**	**Results 1**	**Results 2**
a	1.50	1.414211	1.118032
b	0.75	0.707105	1.118033
d_x	2.00	0.532361	0.532362
d_y	1.00	0.983149	0.983152
$\theta°$	0.53	35.000000	0.070000
A_e	3.53	3.141579	3.926977
Iter		52	18

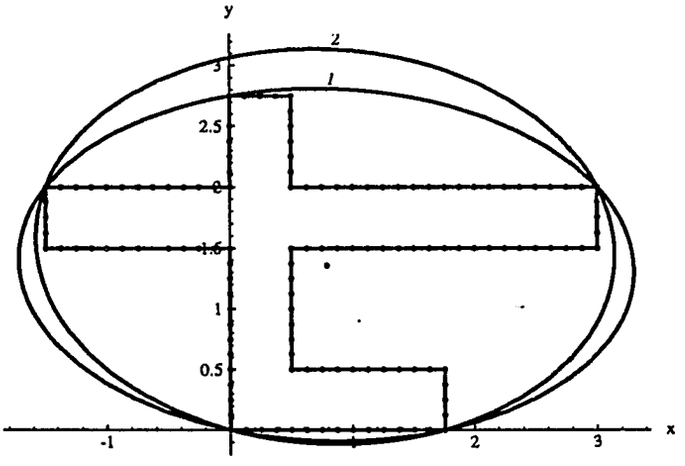


Figure 2.8: Polygon type shape object and approximating ellipses with $N_s = 134$

2.3.3 Example 3

Our next example is an object with more vertices and a more complicated shape. Figure 2.8 shows the object and the samples are taken 0.125 apart. It also shows the approximating ellipses given by the area and approximating error procedures described in Section 2.2.1 and Section 2.2.3, respectively. Table 2.5 shows the initial values and results for both procedures.

Then, fewer samples were used, $N_s = 14$. Figure 2.9 shows the samples, which are the vertices of the object, and the approximating ellipses. Table 2.6 shows the initial values and results for each procedure.

2.3.4 Example 4

Our next example is the same object shown in the previous section but with a non-zero angular orientation. Figure 2.10 shows the object and the approximating ellipses. Table 2.7 shows the initial values and results for both procedures.

Table 2.5: Initial values and results for the parameters in Figure 2.8 with $N_s = 134$

Parameter	Initial	Results 1	Results 2
a	1.75	2.511496	2.363965
b	1.50	1.448127	1.622866
d_x	0.75	0.790091	0.780636
d_y	1.35	1.358540	1.509831
$\theta°$	-35.00	-1.790000	-3.210000
A_e	8.25	11.425864	12.052401
Iter		16	14

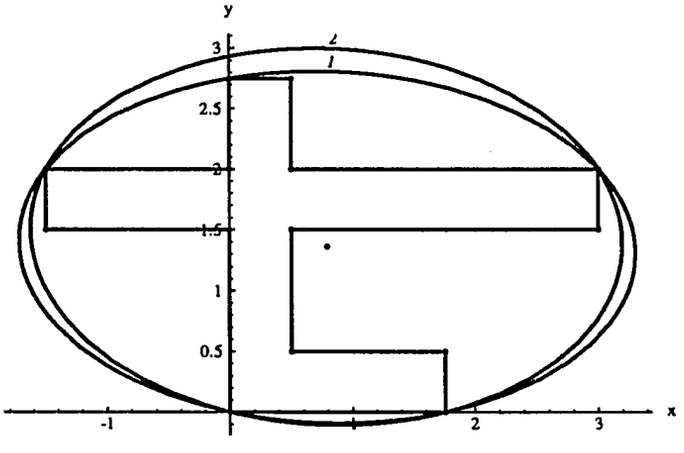


Figure 2.9: Polygon type shape object and approximating ellipses with $N_s = 14$

Table 2.6: Initial values and results for the parameters in Figure 2.9 with $N_s = 14$

Parameter	Initial	Results 1	Results 2
a	1.75	2.511496	2.411788
b	1.50	1.448127	1.550171
d_x	0.75	0.790091	0.784609
d_y	1.35	1.358540	1.446249
$\theta°$	-35.00	-1.790000	-2.530000
A_e	8.25	11.425864	11.745422
Iter		3	8

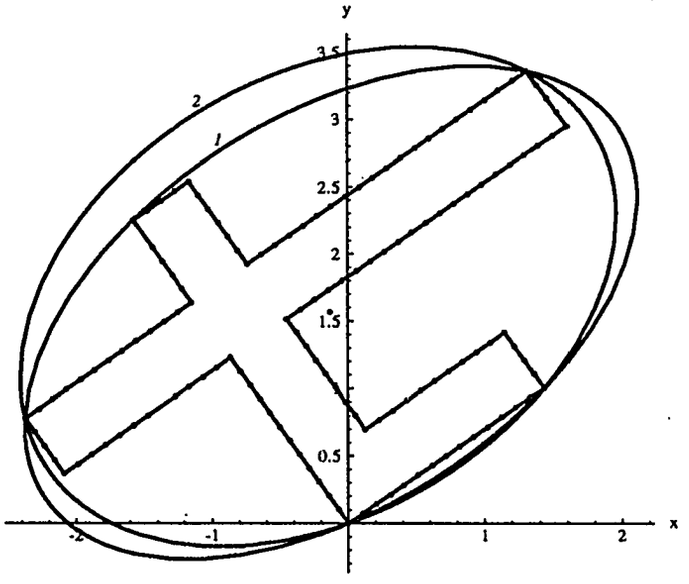


Figure 2.10: Polygon type shape object and approximating ellipses with $N_s = 134$ and non-zero angular orientation

Table 2.7: Initial values and results for the parameters in Figure 2.10 with $N_s = 134$ and non-zero angular orientation

Parameter	Initial	Results 1	Results 2
a	1.75	2.51149	2.363964
b	1.50	1.44813	1.622865
d_x	0.75	-0.13202	-0.226543
d_y	1.35	1.56603	1.684533
$\theta°$	-35	33.21000	31.790000
A_e	8.25	11.42586	12.052389
Iter		10	13

Now, the same object with non-zero angular orientation is used but with the number of samples equal to the number of vertices of the object. Figure 2.11 shows the object and the approximating ellipses and Table 2.8 shows the results.

2.3.5 Example 5

The following example is in 3D. The object chosen was the same object described in Example 3 but with a non-zero value in the z direction. Figure 2.12(a) shows the object and Figure 2.12(b) shows the sorrounding ellipsoid; a detail of both the object and a cutaway of the approximating ellipsoid is shown in Figure 2.12(c). Table 2.9 shows the intial values and results for both procedures.

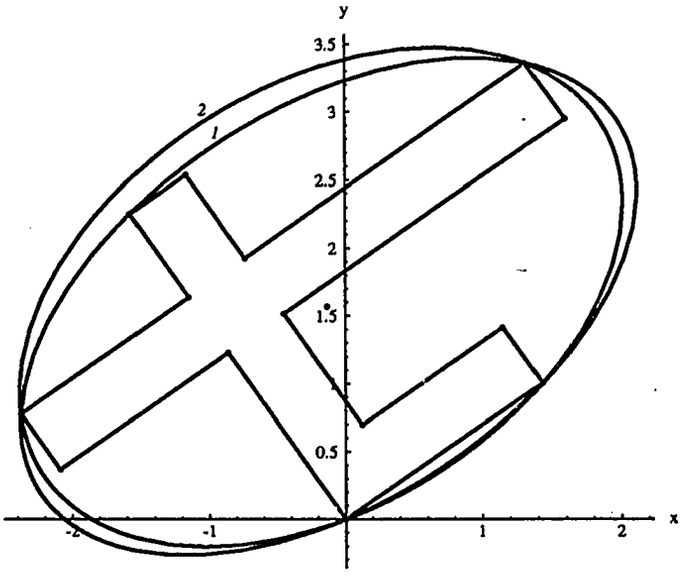
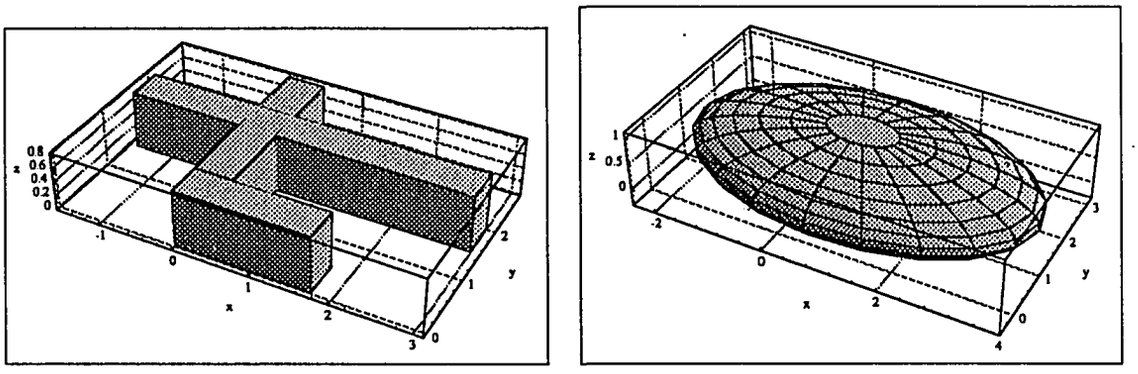


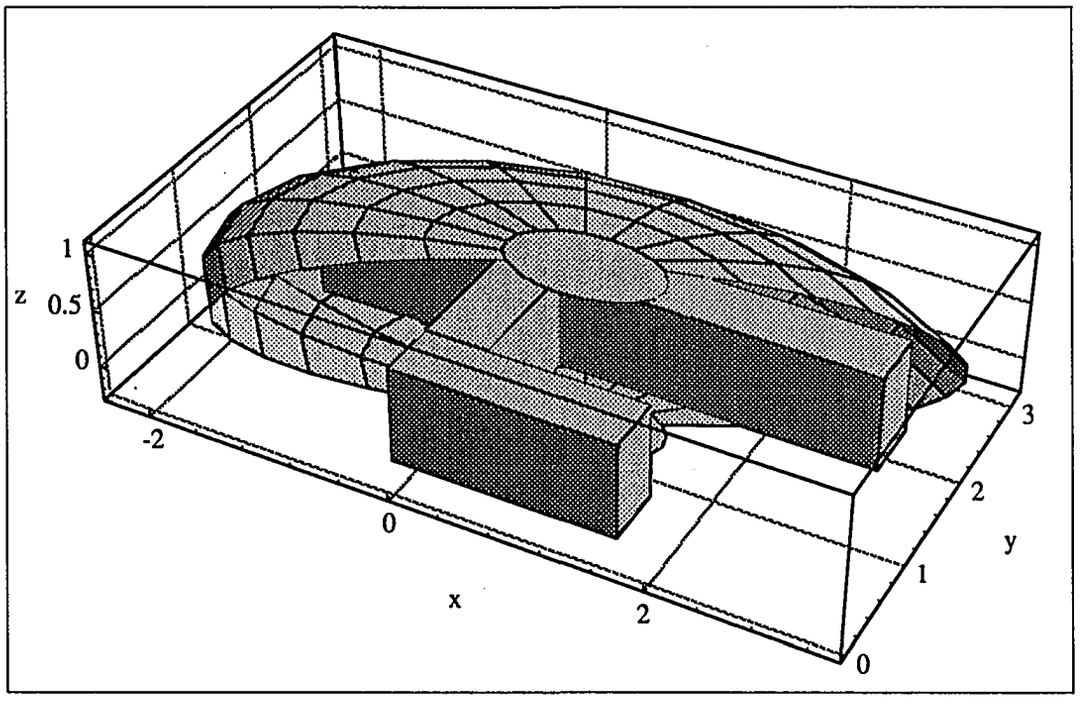
Figure 2.11: Polygon type shape object and approximating ellipses with $N_s = 14$ and non-zero angular orientation

Table 2.8: Initial values and results for the parameters in Figure 2.11 with $N_s = 14$ and non-zero angular orientation

Parameter	**Initial**	**Results 1**	**Results 2**
a	1.75	2.51149	2.41179
b	1.50	1.44813	1.55017
d_x	0.75	-0.13202	-0.18682
d_y	1.35	1.56603	1.63473
$\theta°$	-35	33.21000	32.47000
A_e	8.25	11.42586	11.74542
Iter		10	11



(a) (b)



(c)

Figure 2.12: Object and surrounding ellipsoid

Table 2.9: Initial values and results for the parameters in Figure 2.12 with $N_s = 28$ and non-zero angular orientation

Parameter	Initial	Results 1	Results 2
a	2.50	3.075940	2.543418
b	1.50	1.773585	1.570027
c	0.50	0.692821	1.569959
d_x	0.75	0.790091	0.749869
d_y	1.40	1.358540	1.387587
d_z	0.50	0.400000	0.400151
$\psi°$	0.10	-3.442376	0.005103
$\theta°$	-0.10	0.000000	-28.277574
$\phi°$	0.10	1.654209	0.004346
V_e		15.832136	26.260427
Iter		58	19

3. INTRODUCTION TO B-SPLINES

The origin of the term *spline* goes back to the days before computer graphics, when draftsmen used to locate weights at the data points and then place a flexible wooden ruler, called a *spline*, at the weights, in order to obtain a smooth curve passing through the points. The weights had a protrusion sticking out that fitted into a slot of the spline and had held it in place while allowing it to rotate around the fixed point. It is possible to use the theory of mechanical elasticity and prove that the resulting curve is (approximately) a piecewise cubic polynomial that is continuous and has continuous first and second derivatives. These conditions also assure that the curve has continuous curvature and the discontinuities occur only in the third derivative. Since it is very difficult for the human eye to distinguish the latter, the resulting curve appears completely smooth. If we direct a mechanical motion along a spline, a continuous second derivative implies continuous acceleration and therefore no abrupt changes in force. These two properties make spline curves very desirable for many practical applications.

In many applications where curve fitting is used, one would like to modify parts of the curve without affecting other parts. We say that a scheme has a local property if local modifications do not propagate. Piecewise polynomial functions offer a direct way of achieving local control. We shall discuss such functions in parametric

representation.

3.1 B-spline Basis Functions

The recursive definition known as the Cox-deBoor algorithm is adopted [27, 29, 124]. Let $\mathbf{U} = [u_0, u_1, \ldots, u_i, \ldots, u_m]$ be a non-decreasing sequence of real numbers. The i-th normalized B-spline function of degree k (order $k+1$), denoted by $N_{i,k}(u)$ is defined as follows

$$N_{i,0}(u) = \begin{cases} 1, & u_i \leq u < u_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{3.1a}$$

$$N_{i,k}(u) = \frac{u - u_i}{u_{i+k-1} - u_i} N_{i,k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1,k-1}(u) \tag{3.1b}$$

It is agreed that $0/0 = 0$. The $N_{i,k}(u)$ functions are defined on the entire real line, but our focus is on the interval $u \in [u_0, u_m]$. Note that $N_{i,k}(u)$ is a k-th degree *piecewise* polynomial function. $\mathbf{U}$ is called the knot vector and the u_i values the knots. The interval $[u_i, u_{i+1})$ is called the i-th knot span.

The $N_{i,k}(u)$ functions have the following important properties [29, 124] (see Figure 3.1):

- Non-negativity: $N_{i,k}(u) \geq 0 \quad \forall\, i, k, u.$
- Partition of unity: $\sum_{i=0}^{n} N_{i,k}(u) = 1, \quad \forall\, u \in [u_0, u_m].$
- Local support: $N_{i,k}(u) = 0, \ \forall\ u \notin [u_0, u_m]$. Furthermore, in any given knot span at most $(k+1)$ of the $N_{i,k}(u)$ are non-zero.
- Differentiability: all derivatives of $N_{i,k}(u)$ exist in the interior of a knot span (where it is a polynomial). At a knot $N_{i,k}(u)$ is $(k-j)$ times continuously differentiable, where j is the multiplicity of the knot.

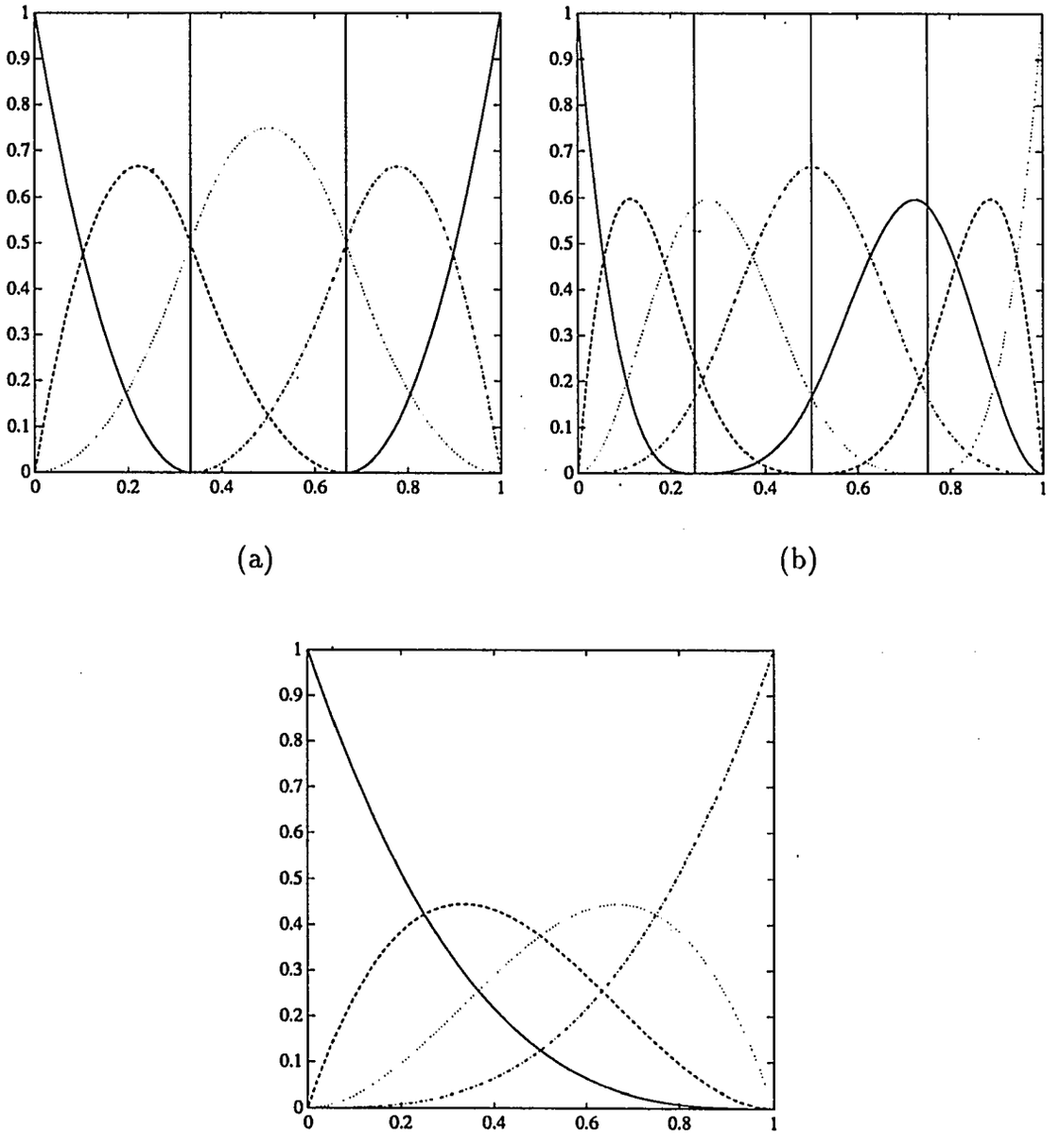


Figure 3.1: B-splines basis functions: (a) quadratic defined by the knot vector [0,0,0,1/3,2/3,1,1,1], (b) cubic defined by [0,0,0,0,1/4,1/2,3/4,1,1,1,1], and (c) cubic defined by [0,0,0,0,1,1,1,1]

- Extrema: except for the case $k = 0$, $N_{i,k}(u)$ attains exactly one maximum value.

The knot vector **U** governs the relationship between parametric and spatial variation, and its entries represent the parameter values at the curve segment joints (knots). The choice of the knot vector **U** clearly influences the shape of the functions defined by Equation 3.1. There are various types of knot vectors. Assume the degree k has been fixed. Then, the knot vector $\mathbf{U} = [u_0, \ldots, u_m]$ is called non-periodic if the first and last knots are repeated with multiplicity $(k+1)$, i.e. $u_0 = u_1 = \ldots = u_k$ and $u_{m-k} = u_{m-k+1} = \ldots = u_m$; furthermore, $u_0 = 0$ and $u_m = 1$. Hence the knot vector has the form

$$\mathbf{U} = [\underbrace{0, \ldots, 0}_{k+1}, u_{k+1}, \ldots, u_{m-k-1}, \underbrace{1, \ldots, 1}_{k+1}] \tag{3.2}$$

A non-periodic B-spline will interpolate (pass through) the first and last control point. If there exists a positive real number d, such that $u_{i+1} - u_i = d$, $\forall\, k \leq i \leq m-k-1$ (equally spaced knots), then **U** is a uniform knot vector; otherwise, it is called non-uniform.

The knot vector

$$[\underbrace{0, \ldots, 0}_{k+1}, \underbrace{1, \ldots, 1}_{k+1}] \tag{3.3}$$

(no interior knots) yields the k-th degree Bernstein basis functions, $B_{i,k}(u)$, defined by

$$N_{i,k}(u) = B_{i,k}(u) = \binom{k}{i} u^i (1-u)^{k-i} \tag{3.4}$$

(see Figure 3.1(c)).

3.2 B-splines

A k-th degree non-rational B-spline curve [124] is defined as follows

$$\mathbf{p}(u) = \sum_{i=0}^{n} N_{i,k}(u)\mathbf{P}_i \tag{3.5}$$

where the $\mathbf{P}_i \in \Re$ or $\Re^2$ or $\Re^3$ are the control points and the $N_{i,k}(u)$ are the k-th degree B-spline basis functions defined by Equation 3.1 and knot vector $\mathbf{U} = [u_0, \ldots, u_i, \ldots, u_m]$. The degree, number of knots, and number of control points are related by the formula

$$m = n + k + 1 \tag{3.6}$$

Note that if the degree of the curve is equal to the number of control points minus one ($k = n$), the general B-spline formulation reduces to a Bézier curve. Also, the parametric derivatives of a B-spline curve (denoted $\mathbf{p}^{(1)}(u) = \mathbf{p}'(u)$, $\mathbf{p}^{(2)}(u) = \mathbf{p}''(u)$, etc.) can be calculated precisely [11, 91].

B-spline curves have a number of useful geometric properties, which follow from definition (3.5) and the analytical properties of the $N_{i,k}(u)$ functions listed in Section 3.1. These properties are (refer to Figure 3.2)

- The multiplicity $(k+1)$ of the end knots yields the following end conditions on the curve:

$$\begin{aligned} &\mathbf{p}(0) = \mathbf{P}_0, && \mathbf{p}(1) = \mathbf{P}_n, \\ &\mathbf{p}'(0) = \frac{k}{u_{k+1}}(\mathbf{P}_1 - \mathbf{P}_0), \quad \text{and} && \mathbf{p}'(1) = \frac{k}{1 - u_{m-k-1}}(\mathbf{P}_n - \mathbf{P}_{n-1}). \end{aligned}$$

- Affine invariance, i.e. a transformation is applied to the curve by applying it to the control points. This follows from the first and second properties of Section 3.1.

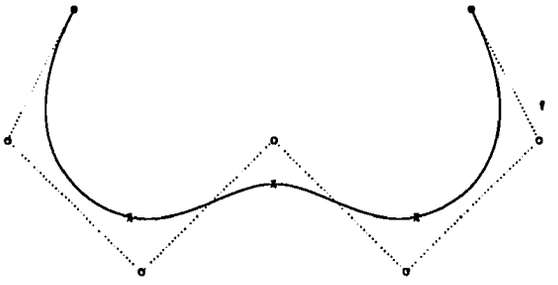


Figure 3.2: Non-rational cubic B-spline curve with its control polygon ('o' points); associated basis functions are given in Figure 3.1(b)

- Strong convex hull property. The curve is contained in the convex hull of its control points. In fact, if u is in the knot span $[u_j, u_{j+1})$, $k \leq j < (m-k-1)$, then $\mathbf{p}(u)$ is in the convex hull of the control points $\mathbf{P}_{j-k}, \ldots, \mathbf{P}_j$.
- The control polygon formed by the $\mathbf{P}_i$ represents a piecewise linear approximation to the curve. This approximation can be improved by knot insertion or by subdivision.
- Local approximation scheme. If a control point is moved, it affects the curve in only $(k+1)$ knot spans (some of which may be degenerate). This follows from the third property of Section 3.1.
- Moving along the curve from $u = 0$ to $u = 1$, the $N_{i,k}(u)$ functions act like switches. As u moves past a knot, one $N_{i,k}(u)$ function (and hence, the corresponding $\mathbf{P}_i$) switches off, and a new one switches on.
- $\mathbf{p}(u)$ is infinitely differentiable at u if u is not a knot (where it is a polynomial). $\mathbf{p}(u)$ is $k-j$ times continuously differentiable at a knot of multiplicity j.
- Variation diminishing property. No plane has more intersections with the curve than with the control polygon.

- A B-spline curve with no interior knots is a Bézier curve (as $N_{i,k}(u) = B_{i,k}(u)$).

3.3 Derivatives of the B-spline and its Basis Functions

The first derivative of a B-spline basis function is given by

$$N_{i,k}^{(1)}(u) = \frac{k}{u_{i+k} - u_i} N_{i,k-1}(u) - \frac{k}{u_{i+k+1} - u_{i+1}} N_{i+1,k-1}(u) \tag{3.7}$$

(proof by induction on k). Repeated differentiation produces the general formula

$$N_{i,k}^{(j)}(u) = k \left[\frac{N_{i,k-1}^{(j-1)}(u)}{u_{i+k} - u_i} - \frac{N_{i+1,k-1}^{(j-1)}(u)}{u_{i+k+1} - u_{i+1}} \right] \tag{3.8}$$

and the j-th derivative of $\mathbf{p}(u)$ (with u fixed) is given by

$$\mathbf{p}^{(j)}(u) = \sum_{i=0}^{n} N_{i,k}^{(j)}(u) \mathbf{P}_i \tag{3.9}$$

Now, instead of fixing u, we want to formally differentiate the k-th degree B-spline curve defined on the knot vector (3.2). Thus,

$$\begin{aligned} \mathbf{p}^{(1)}(u) &= \sum_{i=0}^{n} N_{i,k}^{(1)}(u) \mathbf{P}_i \qquad (3.10) \\ &\doteq \sum_{i=0}^{n} \left(\frac{k}{u_{i-k} - u_i} N_{i,k-1}(u) - \frac{k}{u_{i+k+1} - u_{i+1}} N_{i+1,k-1}(u) \right) \mathbf{P}_i \\ &= k \left(\sum_{i=-1}^{n-1} \left[\frac{\mathbf{P}_{i+1}}{u_{i+k+1} - u_{i+1}} N_{i+1,k-1}(u) \right] - \sum_{i=0}^{n} \left[\frac{\mathbf{P}_i}{u_{i+k+1} - u_{i+1}} N_{i+1,k-1}(u) \right] \right) \end{aligned}$$

or,

$$\mathbf{p}^{(1)}(u) = k \frac{N_{0,k-1}(u)\mathbf{P}_0}{u_k - u_0} + k \sum_{i=0}^{n-1} \left[\frac{\mathbf{P}_{i+1} - \mathbf{P}_i}{u_{i+k+1} - u_{i+1}} N_{i+1,k-1}(u) \right] - k \frac{N_{n+1,k-1}(u)\mathbf{P}_n}{u_{n+k+1} - u_{n+1}} \tag{3.11}$$

Note that the first and last terms evaluate to 0/0, which is 0 by definition. Thus,

$$\mathbf{p}^{(1)}(u) = k \sum_{i=0}^{n-1} \left[N_{i+1,k-1}(u) \frac{\mathbf{P}_{i+1} - \mathbf{P}_i}{u_{i+k+1} - u_{i+1}} \right] = \sum_{i=0}^{n-1} N_{i+1,k-1}(u) \mathbf{Q}_i \tag{3.12}$$

where

$$\mathbf{Q}_i = k\frac{\mathbf{P}_{i+1} - \mathbf{P}_i}{u_{i+k+1} - u_{i+1}} \tag{3.13}$$

Now, let $\mathbf{U}^{(1)}$ be the knot vector obtained by dropping the first and last knots from $\mathbf{U}$, i.e.,

$$\mathbf{U}^{(1)} = [\underbrace{0,\ldots,0}_{k}, u_{k+1}, \ldots, u_{m-k-1}, \underbrace{1,\ldots,1}_{k}] \tag{3.14}$$

($\mathbf{U}^{(1)}$ has $m-1$ knots). Then, it is easy to check that the function $N_{i+1,k-1}(u)$, computed on $\mathbf{U}$, is equal to $N_{i,k-1}(u)$, computed on $\mathbf{U}^{(1)}$. Thus,

$$\mathbf{p}^{(1)}(u) = \sum_{i=0}^{n-1} N_{i,k-1}(u)\mathbf{Q}_i \tag{3.15}$$

where the $N_{i,k-1}(u)$ are computed on $\mathbf{U}^{(1)}$. Hence, $\mathbf{p}^{(1)}(u)$ is a $(k-1)$-th degree B-spline curve. And since $\mathbf{p}^{(1)}(u)$ is a B-spline curve, we can apply this formulation recursively to obtain higher order derivative. Letting $\mathbf{P}_i^{(0)} = \mathbf{P}_i$ and $\mathbf{U} = \mathbf{U}^{(0)}$, we write:

$$\mathbf{p}(u) = \mathbf{p}^{(0)}(u) = \sum_{i=0}^{n} N_{i,k}(u)\mathbf{P}_i^{(0)} \tag{3.16}$$

Then,

$$\mathbf{p}^{(j)}(u) = \sum_{i=0}^{n-j} N_{i,k-j}(u)\mathbf{P}_i^{(j)} \tag{3.17}$$

with

$$\mathbf{P}_i^{(j)} = \begin{cases} \mathbf{P}_i, & j = 0 \\ \frac{k-j+1}{u_{i+k+1} - u_{i-j}}\left(\mathbf{P}_{i+1}^{(j-1)} - \mathbf{P}_i^{(j-1)}\right), & j > 0 \end{cases} \tag{3.18}$$

and

$$\mathbf{U}^{(j)} = [\underbrace{0,\ldots,0}_{k-j+1}, u_{k+1}, \ldots, u_{m-k-1}, \underbrace{1,\ldots,1}_{k-j+1}] \tag{3.19}$$

4. CURVE SYNTHESIS

We may now describe the non-periodic, non-rational, uniform B-spline curve synthesis for our collision-free path planning problem. Consider the synthesis of a cubic B-spline curve with six control points, $n = 6$, i.e.,

$$\mathbf{p}(u) = \sum_{i=0}^{6} N_{i,3}(u)\mathbf{P}_i \tag{4.1}$$

where $N_{i,3}(u)$ are 3^{rd} degree B-spline basis functions and $\mathbf{P}_i \in \Re^2$ are the position vectors of the 7 control points of the curve described by the origin of body M in the vicinity of a single fixed obstacle, F, as shown in Figure 4.1. Suppose, for now, that the orientation of M is fixed. Our problem is to find a curve that:

- passes through the start point, $\mathbf{P}_0$, and goal point,$\mathbf{P}_n$,
- avoids F by at least some clearance, t,
- exhibits a relationship between parametric and spatial variation that is constant across the entire curve, i.e., curve points are equally spaced along the B-spline curve, and
- has manipulator links that avoid F by at least some clearance, t.

We assume an initial configuration such that the interior control points $\mathbf{P}_1^*$-$\mathbf{P}_5^*$ are equally spaced along the line segment $\mathbf{P}_0^*\mathbf{P}_6*$. The curve $\mathbf{p}(u)$ is synthesized by

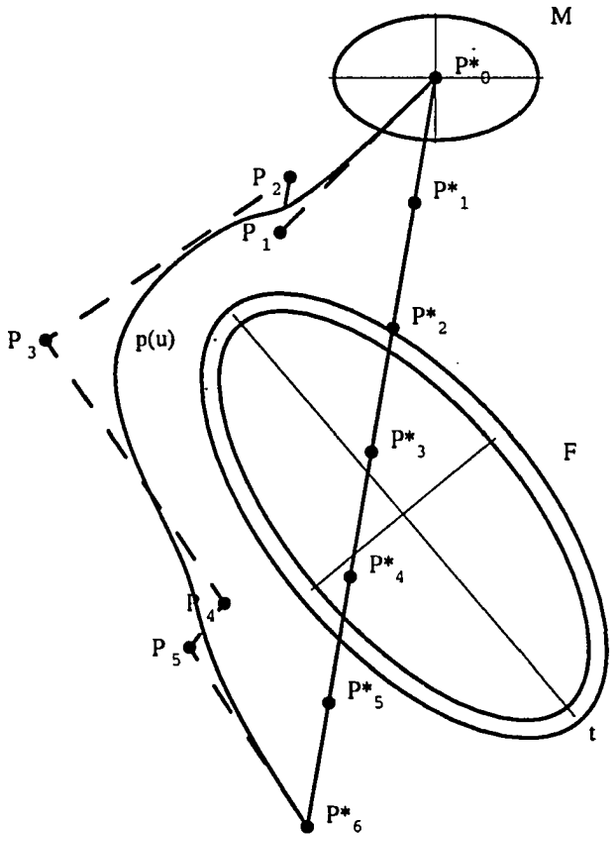


Figure 4.1: A simple path planning problem

finding control points $\mathbf{P}_1$–$\mathbf{P}_5$ such that all constraints are satisfied. This formulation can be extended to more control points and/or obstacles, as described later.

4.1 Cost Function

In order to simultaneously address the global obstacle avoidance and the intrinsic shape characteristics, $(N+1)$ samples of the B-spline curve at constant parametric intervals are obtained. Remember that the B-spline is evaluated in the range $[0,1]$. Then,

$$u_i = u_{i-1} + \frac{1}{N}, \qquad \begin{array}{l} u_0 = 0, \\ i = 1, \ldots, N \end{array} \tag{4.2}$$

Also, four cost components are defined:

- cost due to obstacle proximity, C_d,
- cost due to excessive arc length, C_a,
- cost due to uneven parametric distribution, (curve points not equally spaced along the curve), C_p, and
- cost due to link proximity, C_L.

Then, the sum of the contribution at each sample (point) for each cost component will determine the total cost of the path. That is,

$$C_T = C_d + C_a + C_p + C_L \tag{4.3}$$

The cost components are functions of the curve coordinates $\mathbf{p}(u)$ or its parametric derivative $\mathbf{p}^{(1)}(u)$. The coordinates of the interior control points $\mathbf{P}_1$–$\mathbf{P}_5$ are the

independent variables since $\mathbf{P}_0^*$ and $\mathbf{P}_6^*$ are the start and goal point, respectively. The cost function also depends on the number of parametric samples, N. This input parameter has a trade-off effect in terms of accuracy, convergence, and CPU time.

4.1.1 Obstacle Proximity Cost

The cost due to obstacle proximity, C_d, requires a procedure to determine:

- possible interference between moving object, M, and fixed object, F, and/or
- minimum distance between M and F.

This procedure is described in detail in Section 5.3. However, we can mention that C_d is a function of the distance between M and F which we will denote as $d(\mathbf{p}(u_i), M, F)$. This value will only be computed when there is *no* interference between M and F, i.e. M does *not* intersect F, nor are they tangent. Then, C_d may be defined as

$$C_d = \sum_{i=0}^{N} c_i \tag{4.4}$$

where

$$c_i = \begin{cases} w_{in}, & M \cap F \\ w_{out}\left(1 - d(\mathbf{p}(u_i), M, F)/t\right), & 0 < d(\mathbf{p}(u_i), M, F_k) < t \\ 0, & t \leq d(\mathbf{p}(u_i), M, F) \end{cases} \tag{4.5}$$

here $w_{in} > w_{out} \geq 0$ are constants to penalize more heavily curved points yielding interference between M and F than those in the clearance area. The way we compute the proximity cost when the distance is in $(0, t]$ will tend to pull the curve (and M) away from F. So, we need to compute a cost due to the increase in length of the curve.

4.1.2 Arc Length Cost

The increase in curve length is measured as a deviation from a straight line connecting the start and goal points ($\mathbf{P}_0$ and $\mathbf{P}_6$). A chordal approximation is used to approximate the total length of the curve, i.e.,

$$L_c = \sum_{i=0}^{N-1} \mid \mathbf{p}(u_{i+1}) - \mathbf{p}(u_i) \mid \tag{4.6}$$

Thus,

$$C_a = w_a \left(L_c - \mid \mathbf{P}_6 - \mathbf{P}_0 \mid\right) \tag{4.7}$$

where $w_a \geq 0$ is a constant.

4.1.3 Parametric Cost

Now we need to describe an intrinsic constraint, the parametric distribution, to address the local behavior of the curve since C_d and C_a are global. The parametric cost will try to keep a uniform relationship between parametric and spatial variations, giving as a result a *smooth* curve. In other words, this property will try to drive the curve points toward relatively equal spacing. The cost is defined as

$$C_p = w_p \sum_{i=0}^{N-1} \left| \mid \mathbf{p}(u_{i+1}) - \mathbf{p}(u_i) \mid - \frac{L_c}{N-1} \right| \tag{4.8}$$

where $w_p \geq 0$ is a constant.

4.1.4 Link Proximity Cost

The cost due to manipulator link proximity requires a procedure to determine:

- manipulator joint variables (angles) and link position in terms of the end-effector position and orientation (assume we have N_L links),

- unreachable positions for the manipulator,
- interference between link j, and fixed object, F, and/or,
- minimum distance between link j and F.

In essence, this cost is very similar to cost due to obstacle proximity. However, this cost requires a procedure to solve the inverse kinematics problem for the manipulator in order to compute the cost.

Once we have synthesized the position and orientation for each link, the procedure is exactly the same as the one for proximity cost, that is, link j will replace M, w_{in}^L will replace w_{in}, and w_{out}^L will replace w_{out}. In the other hand, if the position is unreachable, it is handled as an interference between link j and F and a high cost, w_{in}^L, is assigned. The constants w_{in}, w_{out}, w_a, w_p, w_{in}^L, and w_{out}^L determine the importance we want to give to each cost component. These values are heuristically determined by observing the quality of the solution and the computational effort.

4.1.4.1 Inverse Kinematics Problem The problem of inverse kinematics is, in general, difficult to solve since the resulting equations are nonlinear. These equations are much more difficult to solve directly in closed form, and we need to use efficient and systematic techniques that exploit the particular kinematic structure of the manipulator. Whereas the forward kinematics problem always has a unique solution which can be obtained simply by evaluating the forward equations, the inverse kinematics problem may or may not have a solution. Also, if the solution exists, it may or may not be unique. Furthermore, because the forward kinematic equations are in general complicated nonlinear functions of the joint variables, the solutions may be difficult to obtain even when they exist.

In solving the inverse kinematics problem, we are more interested in finding a closed form solution of the equations rather than a numerical solution. Closed form solutions are preferable for two reasons:

- in certain applications, the forward kinematic equations must be solved at a rapid rate, and
- the equations, in general, have multiple solutions.

Having closed form solutions allows one to develop rules for choosing a particular solution from among several.

The practical question of the existence of solutions to the inverse kinematics problem depends on engineering as well as mathematical considerations. We will assume that the given position and orientation is such that at least one solution exists. Once a solution to the mathematical equations is identified, it must be further checked to see whether or not it satisfies all constraints over the ranges of possible joint motions. Thus, a geometric approach was used to solve the inverse kinematics problem for a revolute-revolute planar manipulator to demonstrate the proposed path planning technique.

For example, for a two-link planar manipulator there may be no solution if the given (x, y) coordinates are out of reach of the manipulator. If the given (x, y) coordinates are within the manipulator's reach, there may be two solutions as shown in Figure 4.2(a) (the so called *elbow up* and *elbow down* configurations) or there may be exactly one solution if the manipulator must be fully extended to reach the point. There may even be an infinite number of solutions in some cases [152].

Consider the diagram of Figure 4.2(b). Using the Law of Cosines, θ_2 can be

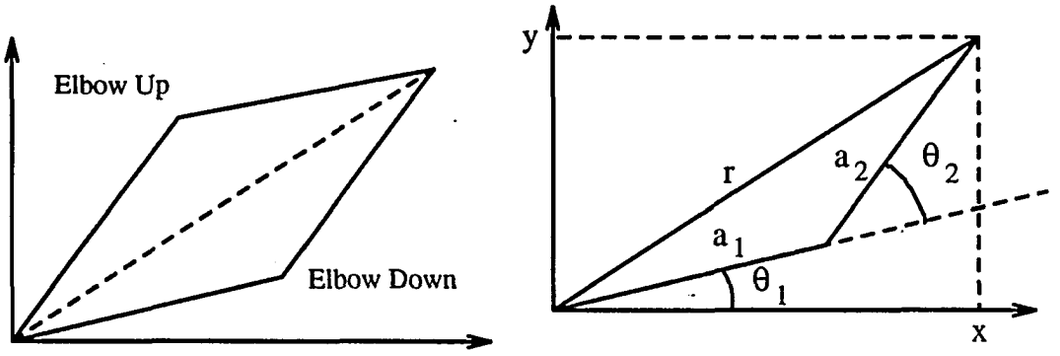


Figure 4.2: Two inverse kinematic solutions and solving for the joint angles of a two-link planar arm

expressed as

$$\cos\theta_2 = \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1a_2} \equiv C_c \tag{4.9}$$

We then determine θ_2 as

$$\theta_2 = \cos^{-1}(C_c) \tag{4.10}$$

However, an alternate way to find θ_2 is to notice that if $\cos(\theta_2)$ is given by Equation 4.9, then $\sin(\theta_2)$ is given by

$$\sin\theta_2 = \pm\sqrt{1 - C_c^2} \equiv C_s \tag{4.11}$$

and, hence, θ_2 can be found by

$$\theta_2 = \tan^{-1}\frac{C_s}{C_c} \tag{4.12}$$

The advantage of this latter approach is that both the elbow-up and elbow-down solutions are found by choosing the positive and negative signs in Equation 4.9, respectively.

Finally, θ_1 is given by

$$\theta_1 = \tan^{-1}\frac{y}{x} - \tan^{-1}\left(\frac{a_2 \sin\theta_2}{a_1 + a_2\cos\theta_2}\right) \tag{4.13}$$

Notice that θ_1 depends on θ_2. This makes sense physically since we would expect a different value for θ_1 depending on which solution is chosen for θ_2.

4.2 Distance Computation

The cost function requires computation of a minimum distance or interference between two ellipses. In order to accomplish this, a constrained minimization problem is formulated. The procedure will be described in 2D, although the extension to 3D is fairly easy; the changes needed to accomplish this in 3D are described in Section 4.2.2.

4.2.1 Distance Computation in 2D

To begin, we will assign an xy-coordinate system to one ellipsoid and an $\hat{x}\hat{y}$-coordinate system to the other. Actually, this is what really happens when we compute the parameters for each enclosing ellipsoid; we use those parameters to rotate and translate them. Now, the objective function will be defined as half the square of the Euclidean norm of the difference between two points, i.e. $\mathbf{r}_1 = [x_1,\, y_1]$ and $\mathbf{r}_2 = [x_2,\, y_2]$, in the xy-coordinate system:

$$\begin{aligned} L(\mathbf{r}_1, \mathbf{r}_2) &= \frac{1}{2}(\mathbf{r}_1 - \mathbf{r}_2)^T(\mathbf{r}_1 - \mathbf{r}_2) && \text{(4.14a)} \\ &= \frac{1}{2}(x_1 - x_2)^2 + \frac{1}{2}(y_1 - y_2)^2 && \text{(4.14b)} \end{aligned}$$

Now, since we need these two points to lie on the perimeter of each ellipsoid, the

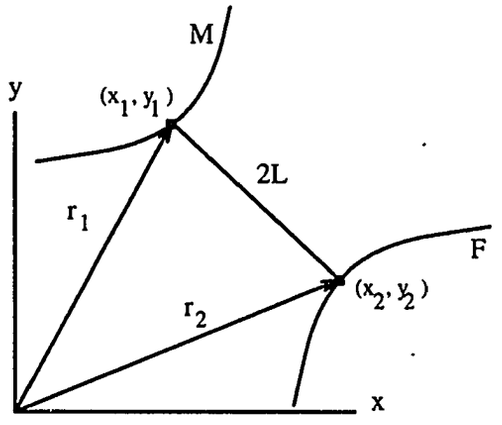


Figure 4.3: Distance between two points

constraints are given by the equation of an ellipse, one for each object:

$$\mathbf{h} = \begin{bmatrix} h_1(\mathbf{r}_1) \\ h_2(\mathbf{r}_2) \end{bmatrix} = \begin{bmatrix} (b_1 x_1)^2 + (a_1 y_1)^2 - (a_1 b_1)^2 \\ (b_2 \hat{x}_2)^2 + (a_2 \hat{y}_2)^2 - (a_2 b_2)^2 \end{bmatrix} = \mathbf{0} \tag{4.15}$$

Since we know the parameters that define each ellipsoid, we can obtain a direct relationship between the two. Assume that the $\tilde{x}\tilde{y}$-coordinate system is the coordinate system in which the path planning problem is carried out, that is, the global reference coordinate system. Then, we would have, for this case, two transformations and rotations:

$$\tilde{\mathbf{r}} = \mathbf{A}_A \mathbf{r}_A + \mathbf{d}_A \tag{4.16a}$$

$$= \mathbf{A}_B \mathbf{r}_B + \mathbf{d}_B \tag{4.16b}$$

where $\mathbf{A}_\cdot$ is the rotation matrix and $\mathbf{d}_\cdot$ is the translation vector. However, we want to have one of these transformed systems as reference. Then, we can solve Equations 4.16 for $\mathbf{r}_A$ or $\mathbf{r}_B$ in order to be able to transform directly from one system to the other. Thus,

$$\mathbf{A}_B \mathbf{r}_B + \mathbf{d}_B = \mathbf{A}_A \mathbf{r}_A + \mathbf{d}_A \tag{4.17}$$

and now we can solve for either $\mathbf{r}_A$ or $\mathbf{r}_B$. We will solve for $\mathbf{r}_B$:

$$\mathbf{r}_B = \mathbf{B}_B \mathbf{r}_A + \mathbf{e}_B \tag{4.18}$$

where

$$\mathbf{B}_B = \mathbf{A}_B^{-1}\mathbf{A}_A \quad \text{and} \quad \mathbf{e}_B = \mathbf{A}_B^{-1}(\mathbf{d}_A - \mathbf{d}_B)$$

This is exactly what we need: a transformation and a rotation of a point in the xy-coordinate system into the $\hat{x}\hat{y}$-coordinate system. Vectors $\mathbf{d}_A$ and $\mathbf{d}_B$ locate the xy and the $\hat{x}\hat{y}$-coordinate system, respectively, and, $\mathbf{A}_A$ and $\mathbf{A}_B$ are the rotation matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ for ellipsoids 1 and 2, respectively. For this case of 2D, matrix $\mathbf{B}_B$ is a rotation matrix given by the relative angle of A-coordinate system with respect to B-coordinate system. That is

$$\hat{\mathbf{r}}_2 = \hat{\mathbf{A}}\mathbf{r}_2 + \hat{\mathbf{d}}_2 \tag{4.19}$$

where

$$\hat{\mathbf{r}}_2 = \begin{bmatrix} \hat{x}_2 \\ \hat{y}_2 \end{bmatrix}, \quad \hat{\mathbf{A}} = \begin{bmatrix} c_\theta & -s_\theta \\ s_\theta & c_\theta \end{bmatrix}, \quad \hat{\mathbf{d}}_2 = \begin{bmatrix} \hat{d}_{x_2} \\ \hat{d}_{y_2} \end{bmatrix}, \quad \text{and} \quad \theta = \theta_1 - \theta_2$$

Now we can proceed to solve our original constrained minimization problem. To accomplish that, we use the Lagrange Multiplier method. The first step is to form the Hamiltonian:

$$H(\mathbf{r}_1, \mathbf{r}_2, \lambda) = L + \lambda^T \mathbf{h} \tag{4.20}$$

where $\lambda = [\lambda_1, \lambda_2]^T$ is an as yet undetermined Lagrange multiplier vector. The necessary conditions for a minimum of L that also satisfies the constraint vector, $\mathbf{h}$,

are given by the gradient of H set equal to zero, $\nabla H^T = \mathbf{0}$, that is

$$\begin{bmatrix} H_{x_1} \\ H_{y_1} \\ H_{x_2} \\ H_{y_2} \\ H_{\lambda_1} \\ H_{\lambda_2} \end{bmatrix} = \begin{bmatrix} x_1 - x_2 + 2\lambda_1 x_1 \\ y_1 - y_2 + 2\lambda_1 y_1 \\ -x_1 + x_2 + 2\lambda_2 \left(b_2 \hat{x}_2 c_\theta - a_2 \hat{y}_2 s_\theta\right) \\ -y_1 + y_2 + 2\lambda_2 \left(b_2 \hat{x}_2 s_\theta + a_2 \hat{y}_2 c_\theta\right) \\ h_1 \\ h_2 \end{bmatrix} = \mathbf{0} \tag{4.21}$$

Here, $H_{(\cdot)} = \partial H / \partial(\cdot)$.

A stationary point of the set of Equations 4.21 would yield a minimum of L and also would satisfy the constraint vector $\mathbf{h}$. We could use Newton's method for solving nonlinear equations. However, it has an unfortunate tendency to wander off into the wild blue yonder if the initial guess is not sufficiently close to the root. An algorithm that combines the rapid local convergence of Newton's method with a globally convergent strategy that will guarantee some progress towards the solution at each iteration will be described in Chapter 5.

4.2.2 Distance Computation in 3D

This section will describe the necessary changes of the objective and constraint functions, shown in Section 4.2.1, in order to accomplish a distance computation in 3D. We will start with the objective function, which in essence is the same: it is half the square of the norm 2 of the difference between two points in $\Re^3$:

$$L(\mathbf{r}_1, \mathbf{r}_2) = \frac{1}{2}(\mathbf{r}_1 - \mathbf{r}_2)^T(\mathbf{r}_1 - \mathbf{r}_2) \tag{4.22a}$$

$$= \frac{1}{2}\left[(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2\right] \tag{4.22b}$$

Thus, the constraints become

$$\mathbf{h} = \begin{bmatrix} h_1(\mathbf{r}_1) \\ h_2(\mathbf{r}_2) \end{bmatrix} = \begin{bmatrix} (b_1c_1x_1)^2 + (a_1c_1y_1)^2 + (a_1b_1z_1)^2 - (a_1b_1c_1)^2 \\ (b_2c_2\hat{x}_2)^2 + (a_2c_2\hat{y}_2)^2 + (a_2b_2\hat{z}_1)^2 - (a_2b_2c_2)^2 \end{bmatrix} = \mathbf{0} \qquad (4.23)$$

The relationship between xy and $\hat{x}\hat{y}$-coordinate systems is given by Equation 4.18. The transformation matrices are given by Equation 2.5 with Euler angles ψ_i, θ_i, and ϕ_i, $i \in \{1, 2\}$. The form of the Hamiltonian remains the same as shown by Equation 4.20. The gradient of H is the one that changes. It is given by

$$\begin{bmatrix} H_{x_1} \\ H_{y_1} \\ H_{z_1} \\ H_{x_2} \\ H_{y_2} \\ H_{z_2} \\ H_{\lambda_1} \\ H_{\lambda_2} \end{bmatrix} = \begin{bmatrix} (1 - 2\lambda_1 b_1 c_1)x_1 - x_2 \\ (1 - 2\lambda_1 a_1 c_1)y_1 - y_2 \\ (1 - 2\lambda_1 a_1 b_1)z_1 - z_2 \\ -x_1 + x_2 + 2\lambda_2(b_2c_2\hat{x}_2\tilde{a}_{1,1} + a_2c_2\hat{y}_2\tilde{a}_{1,2} + a_2b_2\hat{z}_2\tilde{a}_{1,3}) \\ -y_1 + y_2 + 2\lambda_2(b_2c_2\hat{x}_2\tilde{a}_{2,1} + a_2c_2\hat{y}_2\tilde{a}_{2,2} + a_2b_2\hat{z}_2\tilde{a}_{2,3}) \\ -z_1 + z_2 + 2\lambda_2(b_2c_2\hat{x}_2\tilde{a}_{3,1} + a_2c_2\hat{y}_2\tilde{a}_{3,2} + a_2b_2\hat{z}_2\tilde{a}_{3,3}) \\ h_1 \\ h_2 \end{bmatrix} = \mathbf{0} \qquad (4.24)$$

4.3 Orientation of the Moving Object

Although the orientation of the moving object does not contribute directly to the cost function, it does yield a different optimal solution for each case analyzed. Three possible orientation cases are analyzed:

Fixed: The orientation remains the same during the path synthesis. It is an input parameter for the computer program developed to solve our path planning problem.

As a design variable: An initial orientation is provided and every time that the control point coordinates are perturbed, so is the orientation of the moving object. Once perturbed, it remains fixed during the cost function evaluation.

Given by the first derivative of the curve: Every time a new path is computed, so is the slope or Euler angles at each $\mathbf{p}(u_i)$, $i = 0, \ldots, N$. Then, that value is assigned as its orientation angle or Euler angles to M in order to compute the proximity cost for each c_i component.

This last orientation case is computed for 2D as follows: at each curve point, the first derivative curve will have an x and a y component. Then, to obtain the slope at that point, we use the arctan2(η, ξ) function with ξ and η being the x and y components, respectively, of the B-spline first derivative.

For the 3D case, we use a similar procedure as for the 2D case. The first derivative of the B-spline curve will have x, y, and z components (ξ, η, ζ) for each curve point. Then, we compute two of the Euler angles for each curve point with the following relations:

$$\theta = \text{arctan2}\left(\zeta, \sqrt{\xi^2 + \eta^2}\right) \tag{4.25}$$

$$\phi = \text{arctan2}(\eta, \xi) \tag{4.26}$$

The other Euler angle, ψ, is set to zero.

5. PROCEDURE TO FIND DISTANCE

In this chpater we will discribe the procedure used to find the minimum distance and/or interference between two ellipsoids which is required by the cost function to compute the object/manipulator link proximity cost. Our estated path planning problem requires that a moving object, M, should avoid fixed object, F, by at least some clearance, t. Therefore, in order to determine this, we need to compute the minimum distance among objects M and F. The procedure is a more sophisticated implementation of the multidimensional root finding Newton-Raphson method [30]. This method will try to improve on Newton-Raphson's poor global convergence.

5.1 Globally Convergent Newton's Method

A typical problem gives n functional relations to be zeroed, involving variables x_i, $i = 1, 2, \ldots, n$:

$$F_i(x_1, x_2, \ldots, x_n) = 0 \qquad i = 1, 2, \ldots, n. \tag{5.1}$$

We let $\mathbf{x}$ denote the entire vector of values, x_i, and $\mathbf{F}$ denote the entire vector of functions, F_i. In the neighborhood of $\mathbf{x}$, each of the functions F_i can be expanded in a Taylor series

$$F_i(\mathbf{x} + \delta\mathbf{x}) = F_i(\mathbf{x}) + \sum_{j=1}^{n} \frac{\partial F_i}{\partial x_j} \delta x_j + O(\delta \mathbf{x}^2). \tag{5.2}$$

The matrix of partial derivatives appearing in Equation 5.2 is the *Jacobian* matrix, **J**:

$$J_{i,j} \equiv \frac{\partial F_i}{\partial x_j}. \tag{5.3}$$

In matrix notation Equation 5.2 is

$$\mathbf{F}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{F}(\mathbf{x}) + \mathbf{J}\delta\mathbf{x} + O(\delta\mathbf{x}^2). \tag{5.4}$$

By neglecting terms of order $\delta\mathbf{x}^2$ and higher and by setting $\mathbf{F}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{0}$, we obtain a set of linear equations for the corrections of $\delta\mathbf{x}$ that move each function closer to zero simultaneously, namely

$$\mathbf{J}\delta\mathbf{x} = -\mathbf{F} \tag{5.5}$$

yielding

$$\delta\mathbf{x} = -\mathbf{J}^{-1}\mathbf{F}. \tag{5.6}$$

The corrections are then added to the solution vector,

$$\mathbf{x}_{new} = \mathbf{x}_{old} + \delta\mathbf{x} \tag{5.7}$$

A reasonable strategy to decide whether to accept the Newton step $\delta\mathbf{x}$ is to require the step decrease $| \mathbf{F} |^2 = \mathbf{F}^T\mathbf{F}$. This is the same requirement we would impose if we were trying to minimize

$$f = \frac{1}{2}\mathbf{F}^T\mathbf{F} \tag{5.8}$$

Every solution to Equation 5.1 minimizes Equation (5.8), but there may be local minima of Equation (5.8) that are not solutions to Equation (5.1). Thus, simply applying a minimum finding algorithm is *not* a good idea.

Note that the Newton step is a *descent direction* for f:

$$\nabla f \delta\mathbf{x} = (\mathbf{FJ})(-\mathbf{J}^{-1}\mathbf{F}) = -\mathbf{FF} < 0 \tag{5.9}$$

Thus, our strategy is to always try the full Newton step first, because once we are close enough to the solution we will get quadratic convergence. However, we check at each iteration that the proposed step reduces f. If not, we *backtrack* along the Newton direction until we have an acceptable step. Note that this method essentially minimizes f by taking Newton steps designed to bring ∇f to zero. While the method can still occasionally fail by landing on a local minimum of f, this is quite rare in practice. If this happens, the remedy is to try a new starting point.

5.1.1 Line Searches and Backtracking

When we are not close enough to the minimum of f, taking the full Newton step, $\mathbf{p} = \delta\mathbf{x}$, need not decrease the function. We may move too far for the quadratic approximation to be valid. All we are guaranteed is that *initially* f decreases as we move in the Newton direction. So the goal is to move to a new point $\mathbf{x}_{new}$ along the *direction* of the Newton step $\mathbf{p}$, but not necessarily all the way,

$$\mathbf{x}_{new} = \mathbf{x}_{old} + \beta\mathbf{p}, \quad 0 < \beta \leq 1 \tag{5.10}$$

The aim is to find β so that $f(\mathbf{x}_{new})$ has decreased sufficiently. Since $\mathbf{p}$ is always the Newton direction, we first try $\beta = 1$, the full Newton step. This will lead to quadratic convergence when $\mathbf{x}$ is sufficiently close to the solution. However, if $f(\mathbf{x}_{new})$ does not meet our acceptance criteria, we *backtrack* along the Newton direction, trying a smaller value of β, until we find a suitable point. Since the Newton direction is a descent direction, we are guaranteed to decrease f for sufficiently small β.

It is not sufficient to require that $f(\mathbf{x}_{new}) \leq f(\mathbf{x}_{old})$ as the criterion for accepting a step. It can fail to converge to a minimum of f in one of two ways. First, it is possible to construct steps satisfying this criterion with f decreasing too slowly

relative to the step lengths. Second, one can have a sequence where the steps lengths are too small relative to the initial rate of decrease of f.

A simple way to fix the first problem is to require the *average* rate of decrease of f to be at least some fraction, α, of the *initial* rate of decrease of $\nabla f \cdot \mathbf{p}$:

$$f(\mathbf{x}_{new}) \leq f(\mathbf{x}_{old}) + \alpha \nabla f \cdot (\mathbf{x}_{new} - \mathbf{x}_{old}) \tag{5.11}$$

Here the parameter α satisfies $0 < \alpha < 1$. We can get away with quite small values of α ($\alpha = 10^{-4}$ is a good choice).

The second problem can be fixed by requiring the rate of decrease of f at $\mathbf{x}_{new}$ to be greater than some fraction, β, of the rate of decrease of f at $\mathbf{x}_{old}$. In practice, we will not need to impose this second constraint because our backtracking algorithm will have a built-in cutoff to avoid taking steps that are too small.

Here is the strategy for a practical backtracking routine. Define

$$g(\beta) \equiv f(\mathbf{x}_{old} + \beta \mathbf{p}) \tag{5.12}$$

so that

$$g'(\beta) = \nabla f \cdot \mathbf{p} \tag{5.13}$$

If we need to backtrack, then we model g with the most current information we have and choose β to minimize the model. We start with $g(0)$ and $g'(0)$ available. The first step is always the Newton step, $\beta = 1$. If this step is not acceptable, we have available $g(1)$ as well. We can therefore model $g(\beta)$ as a quadratic:

$$g(\beta) \simeq [g(1) - g(0) - g'(0)]\,\beta^2 + g'(0)\beta + g(0) \tag{5.14}$$

Taking the derivative of this quadratic, we find that it is a minimum when

$$\beta = \frac{-g'(0)}{2\,[g(1) - g(0) - g'(0)]} \tag{5.15}$$

Since the Newton step failed, we can show that $\beta \lesssim \frac{1}{2}$ for small α. However, we need to guard against too small a value of β. We set $\beta_{\min} = 0.1$.

On second and subsequent backtracks, we model g as a cubic in β, using the previous value, $g(\beta_1)$, and the second most recent value, $g(\beta_2)$,

$$g(\beta) = k_1\beta^3 + k_2\beta^2 + g'(0)\beta + g(0) \tag{5.16}$$

Requiring this expression to give the correct values of g at β_1 and β_2 gives two equations that can be solved for the coefficients k_1 and k_2:

$$\begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \frac{1}{\beta_1 - \beta_2} \begin{bmatrix} \frac{1}{\beta_1^2} & \frac{-1}{\beta_2^2} \\ \frac{-\beta_2}{\beta_1^2} & \frac{\beta_1}{\beta_2^2} \end{bmatrix} \begin{bmatrix} g(\beta_1) - g'(0)\beta_1 - g(0) \\ g(\beta_2) - g'(0)\beta_2 - g(0) \end{bmatrix} \tag{5.17}$$

The minimum of Equation 5.16 is at:

$$\beta = \frac{-k_2 + \sqrt{k_2 - 3k_1 g'(0)}}{3k_1} \tag{5.18}$$

We require β to lie between $\beta_{\max} = 0.5\beta_1$ and $\beta_{\min} = 0.1\beta_1$.

5.2 Examples

For the following examples, two figures will be shown for each one. One will show the two ellipsoids and where the point lie which give minimum distance or interference. The other figure will show how those points are reached. The initial values are marked by a star.

The initial values to start the search are chosen as

$$\zeta_1 = \zeta_2 = \frac{\zeta_{1,2}}{3} + R \min_{i \in \{1,2\}} (b_i) \tag{5.19}$$

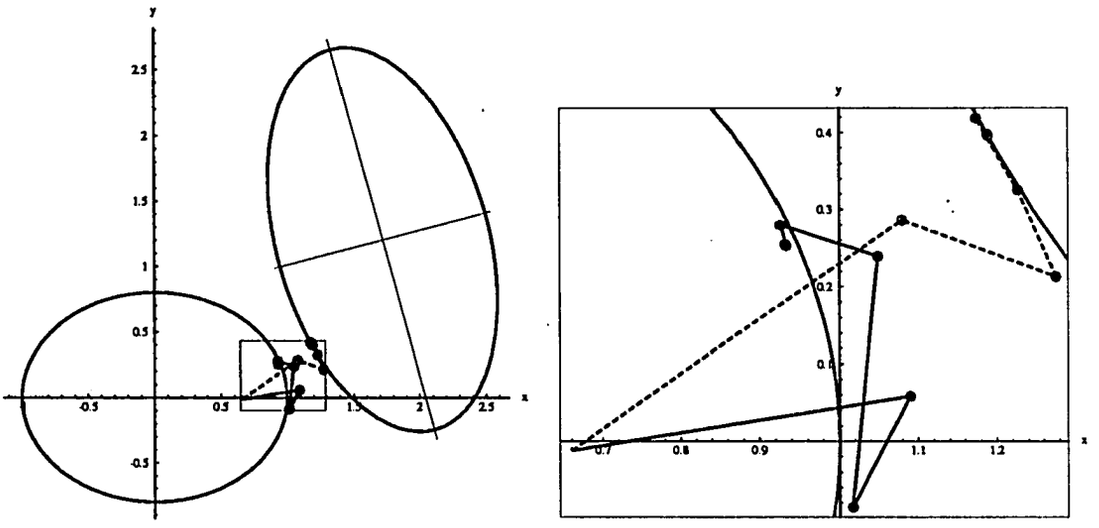
Table 5.1: Parameter values for Figure 5.1

	Ellipse	
Parameter	**1**	**2**
a	1.0	1.50
b	0.7	0.80
$\theta°$	0.0	105.00
(d_x, d_y)	(0,0)	(1.72, 1.20)
$\mathbf{r}_1$	(0.931462, 0.254693)	
$\mathbf{r}_2$	(1.187744, 0.397701)	
distance		0.2935
seed		12345678.0000

where $\zeta_{(\cdot)}$ is a component of the coordinate system, $\zeta_{1,2}$ is the difference from origin 1 to origin 2 in the ζ direction, and R is a random number between (0,1). This relationship was chosen because the solution will be located in the direction of a vector from origin 1 pointing to origin 2.

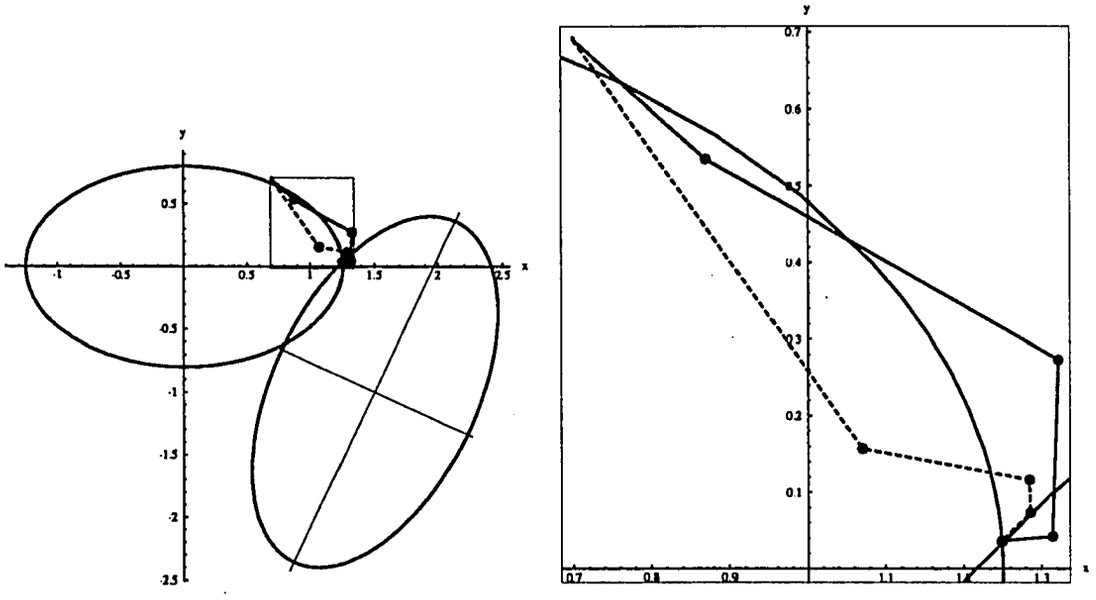
5.2.1 2 Dimensions

Our first example shows an ellipse in the vicinity of another one, see Figure 5.1. Table 5.1 shows the parameter values for each ellipse and the location of the points yielding a minimum distance. The next example, Figure 5.2 shows two other ellipses that exhibit an interference. Table 5.2 shows the parameter values for each ellipse and the location of final points.



(a) (b)

Figure 5.1: Location of points yielding minimum distance between two ellipses: (a) complete figure and (b) zoom of the location



(a) (b)

Figure 5.2: Location of points yielding interference between two ellipses: (a) complete figure and (b) zoom of the location

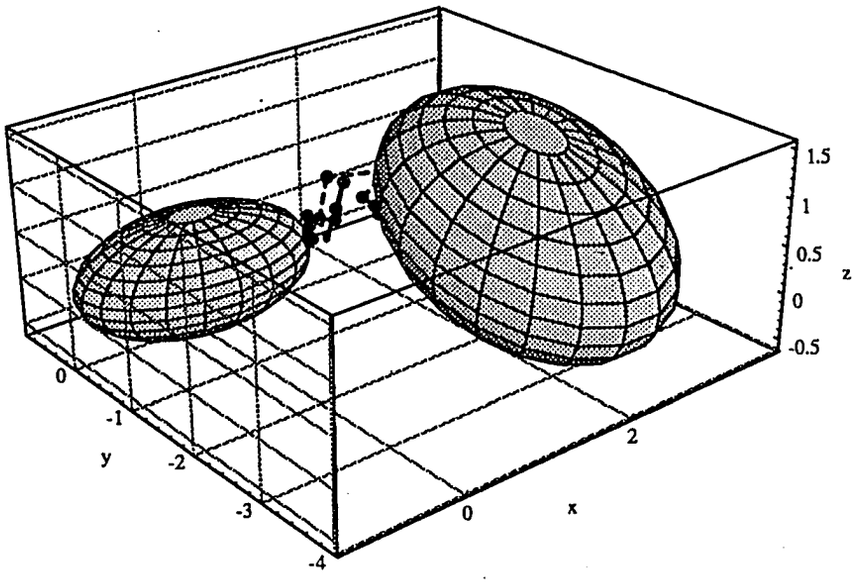
Table 5.2: Paremeter values for Figure 5.2

	Ellipse	
Parameter	**1**	**2**
a	1.25	1.50
b	0.70	0.80
$\theta°$	0.00	65.00
(d_x, d_y)	(0,0)	(1.50,-1.00)
$\mathbf{r}_1$	(1.248397, 0.035438)	
$\mathbf{r}_2$	(1.248397, 0.035438)	
distance		0
seed		12345678

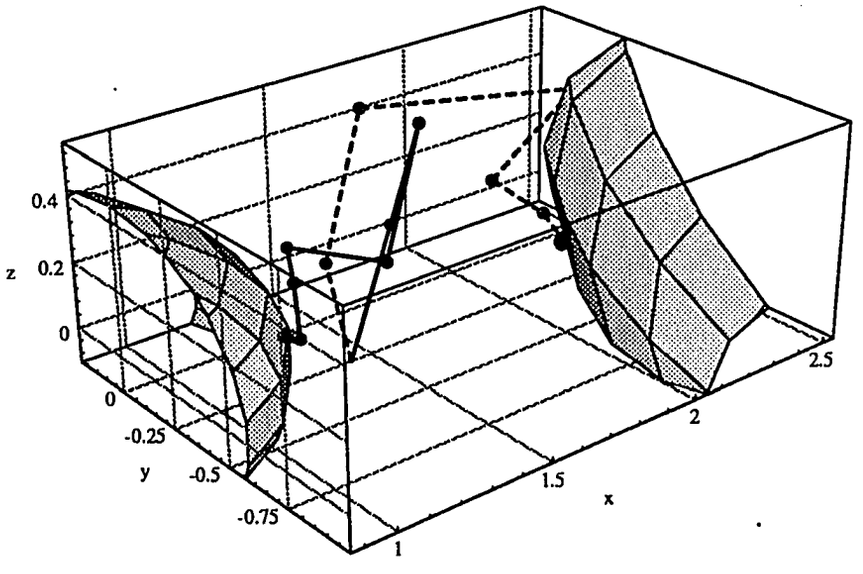
5.2.2 3 Dimensions

Figure 5.3 shows two ellipsoids and the steps taken to reach the points, on the surface of the ellipsoids, that yield minimum distance between the two ellipsoids. Table 5.3 shows the parameter values for each ellipsoid and the location of the points which yield minimum distance. The initial values for points 1 and 2 are equal and are chosen the same way explained for 2D in the previous section.

The following example presents two ellipsoids when there exists interference between them. Figure 5.4 shows the ellipsoids and the steps taken to reach the points yielding minimum distance between the ellipsoids. Table 5.4 shows the parameter values for each ellipsoid, the points yielding minimum distance, $\mathbf{r}_1$ and $\mathbf{r}_2$, and the corresponding distance which in this case is zero because the ellipsoids are intersecting with each other.



(a)



(b)

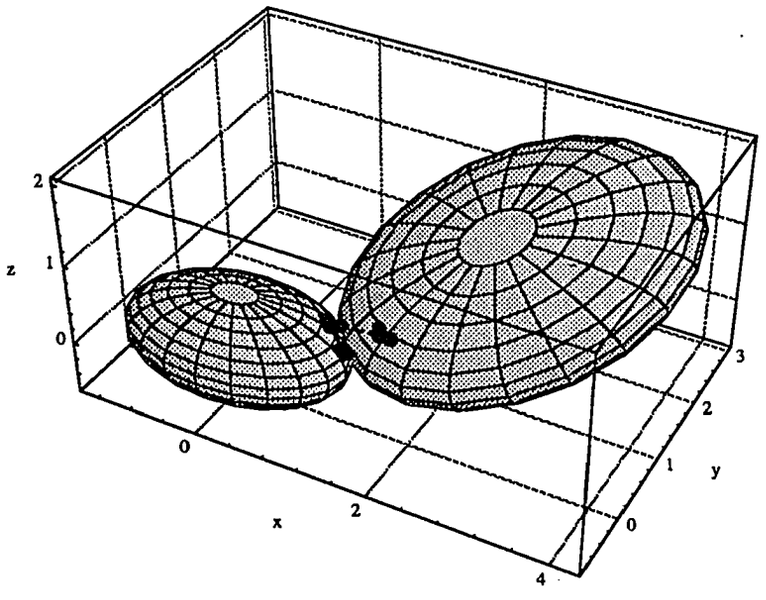
Figure 5.3: Location of points yielding minimum distance between two ellipsoids: (a) complete figure and (b) zoom of the location

Table 5.3: Paremeter values for Figure 5.3

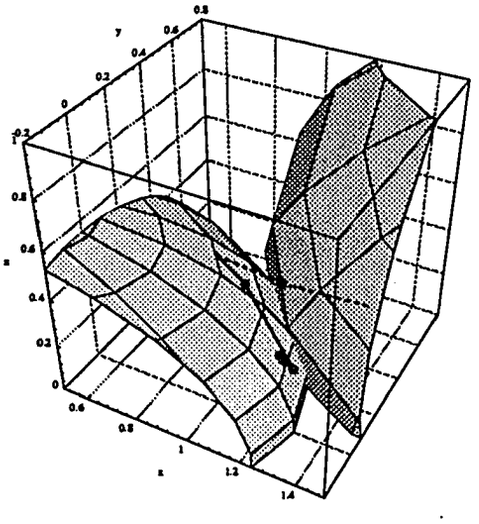
	Ellipse	
Parameter	**1**	**2**
a	1.3	2.0
b	0.8	1.3
c	0.6	1.0
$(\psi, \theta, \phi)°$	(0,0,0)	(0,10,75)
(d_x, d_y, d_z)	(0,0,0)	(2.5,-2,0.5)
$\mathbf{r}_1$	(1.146176, -0.345219, 0.114533)	
$\mathbf{r}_2$	(1.706160, -0.790598, 0.377221)	
distance		0.7622
seed		12345678.0000

Table 5.4: Paremeter values for Figure 5.4

	Ellipse	
Parameter	**1**	**2**
a	1.3	2.0
b	0.8	1.3
c	0.6	1.0
$(\psi, \theta, \phi)°$	(0,0,0)	(0,35,45)
(d_x, d_y, d_z)	(0,0,0)	(2.5,1.5,0.75)
$\mathbf{r}_1$	(0.9177, 0.4684, 0.2699)	
$\mathbf{r}_2$	(0.9177, 0.4684, 0.2699)	
distance		0
seed		81726354



(a)



(b)

Figure 5.4: Location of points yielding interference between two ellipsoids: (a) complete figure and (b) zoom of the location

5.3 Procedure to Find Interference/Distance

In order to reduce superfluous distance calculations, we test first the distance, d_o, between the origin of the moving object, M, located at $\mathbf{p}(u_i)$, and the origin of the k-th obstacle, F_k:

1. If $d_o > (a_M + a_F + t)$, there is no interference and no need to compute a minimum distance, $c_i = 0$. Test the next object.
2. If $d_o \leq (b_M + b_F)$, M intersects or is tangent to F, $c_i = w_{in}$. Test the next object.
3. If none of the above is true, start the procedure to find the minimum distance between two ellipsoids.

Here a_M and b_M are ellipsoids parameters defining the moving object, M, a_F and b_F are ellipsoids parameters defining the fixed object F, and c_i is the obstacle proximity cost component at point $\mathbf{p}(u_i)$ of the path.

Once the distance between an ellipsoid in the vicinity of another has been computed, as described in Section 4.2.1, Equation 4.5 is used to determine the contribution to the cost function for that particular position, $\mathbf{p}(u_i)$. The previous procedure is carried out for all points, N, in the path and for all fixed objects (obstacles), N_o.

6. SIMULATED ANNEALING ALGORITHM

The *simulated annealing algorithm* is a technique that has attracted significant attention as suitable for optimization problems of large scale, especially ones where a desired global extremum is hidden among many, poorer, local extrema [40, 42, 44, 47, 48, 86, 128, 140]. For practical purposes, simulated annealing has effectively "solved" the famous traveling salesman problem of finding the shortest cyclical itinerary for a traveling salesman who must visit each of some number of cities in turn [128]. The method has also been used successfully for designing complex integrated circuits [18, 31, 45, 182]. The arrangement of several hundred thousand circuit elements on a tiny silicon substrate is optimized so as to minimize interference among their connecting wires. Surprisingly, the implementation of the algorithm is relatively simple.

Notice that the two applications cited are both examples of *combinatorial optimization*. There is an objective function to be minimized, as usual; but the space over which that function is defined is not simply the n-dimensional space of n continuously variable parameters. Rather, it is a discrete, but very large, configuration space. The number of elements in the configuration space is factorially large, so that they cannot be explored exhaustively. Furthermore, since the set is discrete, we are deprived of any notion of "continuing downhill in a favorable direction." The concept of "direction" may not have any meaning in the configuration space [86].

This chapter describes the algorithm that was used for our path planning and obstacle avoidance problem and its differences from the "standard" procedure.

6.1 The Algorithm

In its original form, the simulated annealing algorithm is based on the analogy between the simulation of the annealing of solids and the problem of solving large combinatorial optimization problems [76]. For this reason the algorithm became known as "simulated annealing." In condensed matter physics, annealing denotes a physical process in which a solid in a *heat bath* is heated up by increasing the temperature of the heat bath to a maximum value at which all particles of the solid randomly arrange themselves in the liquid phase, followed by cooling by slowly lowering the temperature of the heat bath. In this way, all particles arrange themselves in the low energy ground state of a corresponding lattice, provided the maximum temperature is sufficiently high and the cooling is carried out sufficiently slowly. Starting off at the maximum value of the temperature, the cooling phase of the annealing process can be described as follows. At each temperature value, T, the solid is allowed to reach *thermal equilibrium*, characterized by a probability of being in a state with energy E given by the *Boltzmann distribution*:

$$P(E) = \frac{1}{Z(T)} \exp\left(-\frac{E}{k_B T}\right), \tag{6.1}$$

where $Z(T)$ is a normalization factor known as the *partition function* which depends on the temperature T, and k_B is the *Boltzmann constant*. The factor

$$\exp\left(-\frac{E}{k_B T}\right)$$

is know as the *Boltzmann factor*. As the temperature decreases, the Boltzmann distribution concentrates on the states with lowest energy and finally, when the temperature approaches zero, only the minimum energy states have a non-zero probability of occurrence. However, it is well known that if the cooling is too rapid, i.e. if the solid is not allowed to reach thermal equilibrium for each temperature value, defects can be "frozen" into the solid and metastable amorphous structures can be reached rather than the low energy crystalline lattice structure. Furthermore, in a process known in condensed matter physics as *quenching*, the temperature of the heat bath is lowered instantaneously which results again in a freezing of the particles in the solid into one of the metastable amorphous structures.

To simulate the evolution to thermal equilibrium of a solid for a fixed value of the temperature, T, Metropolis *et al.* [106] proposed a *Monte Carlo method* which generates sequences of states of the solid in the following way. Given the current state of the solid characterized by the positions of its particles, a small perturbation, randomly generated, is applied by a small displacement of a randomly chosen particle. If the difference in energy, ΔE, between the current state and the slightly perturbed one is *negative*, i.e. if the perturbation results in a lower energy for the solid, then the process is continued with the new state. If $\Delta E \geq 0$, then the probability of acceptance of the perturbed state is given by

$$\exp\left(-\frac{\Delta E}{k_B T}\right).$$

This acceptance rule for new states is referred to as the *Metropolis criterion*. Following this criterion, the system eventually evolves into thermal equilibrium, i.e. after a large number of perturbations, using the aforementioned acceptance criterion, the probability distribution of the states approaches the Boltzmann distribution, given by

Equation 6.1. In statistical mechanics this Monte Carlo procedure, which is known as the *Metropolis algorithm*, is a well-known method used to estimate averages or integrals by means of random sampling techniques.

The Metropolis algorithm can also be used to generate sequences of configurations of a combinatorial optimization problem. In that case, the configurations assume the role of the states of a solid while the cost function, C, and the *control parameter*, c, take the roles of energy and temperature, respectively. The simulated annealing algorithm can now be viewed as a sequence of Metropolis algorithms evaluated at a sequence of decreasing values of the control parameter. It can thus be described as follows. Initially, the control parameter is given a high value and a sequence of configurations of the combinatorial optimization problem is generated. As in the iterative improvement algorithm, a generation mechanism is defined, so that, given a configuration i, another configuration j can be obtained by choosing at random an element from the neighborhood of i. The latter corresponds to the small perturbation in the Metropolis algorithm. Let $\Delta C = C_j - C_i$. Then the probability of configuration j to be the next configuration in the sequence is given by 1, if $\Delta C \leq 0$, and by $\exp(-\Delta C/c)$, if $\Delta C > 0$ (the Metropolis criterion). Thus, there is a non-zero probability of *continuing* with a configuration with higher cost than the current configuration. This process is continued until the equilibrium is reached, that is, until the probability distribution of the configuration approaches the Boltzmann distribution.

The control parameter is then lowered in steps, with the system being allowed to approach equilibrium for each step by generating a sequence of configurations in the previously described way. The algorithm is terminated for some small value

of c, for which virtually no deteriorations are accepted any more. The final "frozen" configuration is then taken as the solution of the problem at hand.

Comparing iterative improvement and simulated annealing, it is apparent that the situation where the control parameter in the simulated annealing algorithm is set to zero corresponds to a version of iterative improvement (it is not iterative improvement *per se* because in an iterative improvement approach the neighboring configurations are not necessarily examined in a random order). In the analogy with condensed matter physics, this corresponds to the previously mentioned quenching process. On the contrary, simulated annealing is a generalization of the iterative improvement in that it accepts, with non-zero but gradually decreasing probability, deteriorations in the cost function. However, it is not clear whether it performs better than repeated application of iterative improvement (for a number of different initial configurations). Both algorithms converge asymptotically to a globally minimal configuration of the problem at hand. Lundy and Mees [104] showed for a certain problem that simulated annealing performs better than repeated applications of iterative improvement.

6.2 The Implementation

Given a cost function, $C(\mathbf{z})$, and an initial (solution) state, $\mathbf{z}_0$, the algorithm seeks to improve the current solution by randomly perturbing $\mathbf{z}_0$. The Metropolis algorithm was used for acceptance/rejection of the new state at a given temperature T. The steps in the procedure can be summarized as follows:

1. Randomly perturb $\mathbf{z}$ and calculate the corresponding change in cost, ΔC.
2. If $\Delta C < 0$, accept the state.

3. If $\Delta C > 0$, accept the state with probability

$$P(\Delta C) = \exp\left(\frac{-\Delta C}{T}\right),$$

this represents the core or "inner loop" of the SA algorithm. The acceptance criterion is implemented by generating a random number, $R \in [0, 1]$ and comparing it to $P(\Delta C)$. If $R < P(\Delta C)$, then the new state is accepted. For any given temperature, the inner loop must proceed long enough for the system to reach steady state [76].

The "outer loop" of the algorithm is referred to as the cooling schedule, and specifies the equation by which the temperature is decreased. The algorithm terminates when the cost function remains approximately unchanged, for example, for N_{out} consecutive outer loop iterations.

Any implementation of simulated annealing generally requires four component parts:

1. a problem configuration (domain over which the solution will be sought),
2. a neighborhood definition (which governs the nature and magnitude of allowable perturbations),
3. a cost function, and
4. a cooling schedule (which controls both the rate of temperature decrement and the number of inner loop iterations).

The domain for our problem is simply the real plane in 2D or 3D; control points ($\mathbf{P}_i$) are allowed to take any value in $\Re^2$ or $\Re^3$. The cost function was described in detail in Section 4.1. The neighborhood function used for this study is the same used by Malhotra, *et al.* (1991) which is modeled as an ϵ ball in 2D and

a sphere in 3D around each control point. All free control points of the curve must be perturbed at each step of the SA algorithm. To determine a perturbation for any given control point, four random numbers are generated. Two of them are used to specify the magnitude of the perturbation and the other two to determine the sign of the perturbation ($R \in [0,1]$). To reduce the size of the allowable perturbations with temperature, thus ensuring more local perturbations at low temperature, the following limiting function [32] was used:

$$\epsilon = \epsilon_{max} \left(\frac{\log(T - T_f)}{\log(T_0 - T_f)} \right) \tag{6.2}$$

where ϵ_{max} is an input parameter (or a percent of the linear distance between initial and final curve points), and T, T_0, T_f are the current, initial and final temperatures, respectively. In our study, temperature is simply a high value (of energy).

The cooling schedule is one of the most critical aspects of the SA algorithm. Many variants of the SA algorithm have been presented since its 1982 introduction. Two general classes of this technique were distinguished by VanLaarhoven and Aarts (1998):

Class A Those with a variable number of inner loop iterations (Markov chain length) and fixed temperature decrement, and

Class B Those with a fixed number of inner loop iterations and variable temperature decrement.

However, based on results presented by Malhotra, *et al.* (1991) , a *hybrid* cooling schedule in which both the temperature and the inner loop criterion vary continuously through the annealing process [32, 38] is used. The outer loop behaves nominally as

a constant decrement factor,

$$T_{i+1} = \alpha T_i \tag{6.3}$$

where $\alpha = 0.90$. The temperature throughout the inner loop is allowed to vary proportionally with the current optimal value of the cost function. So, denoting the inner loop index as j, the temperature is modified when a state is accepted, i.e.,

$$T_j = \frac{C_j}{C_{last}} T_{last} \tag{6.4}$$

where C_{last} and T_{last} are the cost and temperature associated with the last accepted state. Note that at high temperatures, a high percentage of states are accepted, so the temperature can fluctuate by a substantial magnitude within the inner loop. By eliminating the necessity to atain equilibrium at high values of temperature [38], this modification reduces computational effort.

A criterion for the inner loop, described by Devadas and Newton (1987) which is based on the number of degrees of freedom of the system and the current temperature, was employed. Since each coordinate direction can be perturbed in both a positive and negative sense, it was considered that the system has two degress of freedom per interior control point coordinate direction. Devadas and Newton observed that at higher temperatures, equilibrium is attained faster, that is, in a fewer number of states. So, they introduce a function to gradually increase the number of states attempted in the inner loop at each temperature. They empirically found that if the number of states per fundamental unit (in our case degree of freedom) varied from about two at high temperatures to about ten at low temperatures, good solutions were obtained without excessive inner loop iterations at higher temperatures. Thus,

the following function was used to determine the number of inner loop iterations,

$$N_{in} = N_{dof}\left[2 + 8\left(1 - \frac{\log(T - T_f)}{\log(T_0 - T_f)}\right)\right] \tag{6.5}$$

where N_{dof} is the number of degrees of freedom in the system. It has been found that this hybrid cooling schedule, which combines the variable temperature scheme of Elperin with the inner loop criterion control of Devadas and Newton, holds for this work too and is an ideal simulated annealing formulation for the collision-free path planning problem.

The initial temperature must be chosen such that the system has sufficient energy to visit the entire solution space. An approach is to simply select an arbitrary temperature and attempt a number of state transitions. The system is sufficiently melted if a large percentage (e.g., 80%) of the state transitions are accepted. If the initial guess for the temperature yields less than this percentage, T_0 can be scaled linearly and the process repeated. An initial guess which yields excessive acceptance (e.g., >95%) is also counter productive, indicating too much energy in the system, so T_0 is scaled accordingly to reach the 80% acceptance level. The algorithm will proceed to a reasonable solution even if there is excessive energy in the system. It is simply less computationally efficient than a solution that starts from a lower temperature. Besides the stopping criterion mentioned above, which indicates convergence to a global minimum, the algorithm is also terminated by setting a final temperature or an upper bound on the number of outer loop iterations, N_{out}. The latter was chosen for our formulation and calcuate the final temperature, T_f, as,

$$T_f = \alpha^{N_{out}} T_0 \tag{6.6}$$

Finally, the feasibility and performance of any simulated annealing implementation depends crucially on a robust random number generator. To enhance code portability and the overall quality of our random numbers, the simple Lehmer generator algorithm presented by Park and Miller (1988) was implemented giving excellent results.

Figure 6.1 shows a flowchart of the computer program implementation of the simulated annealing and its cost function.

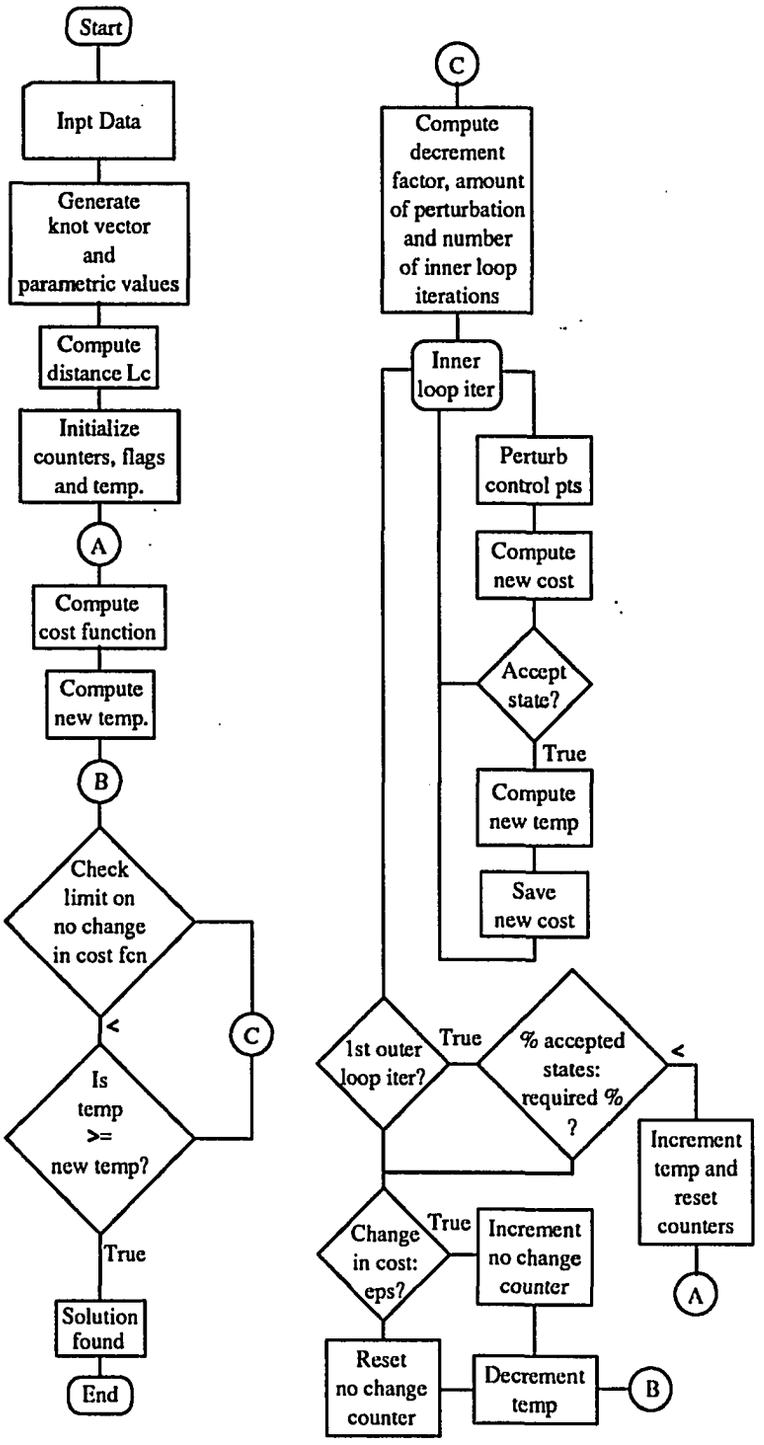


Figure 6.1: Computer program implementation of the simulated annealing algorithm

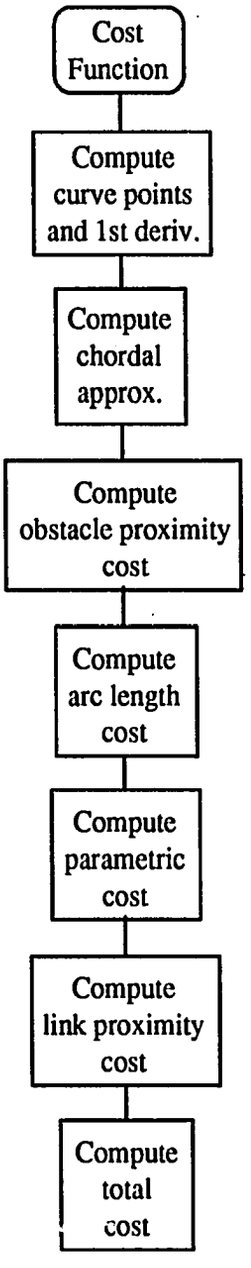


Figure 6.1 (Continued)

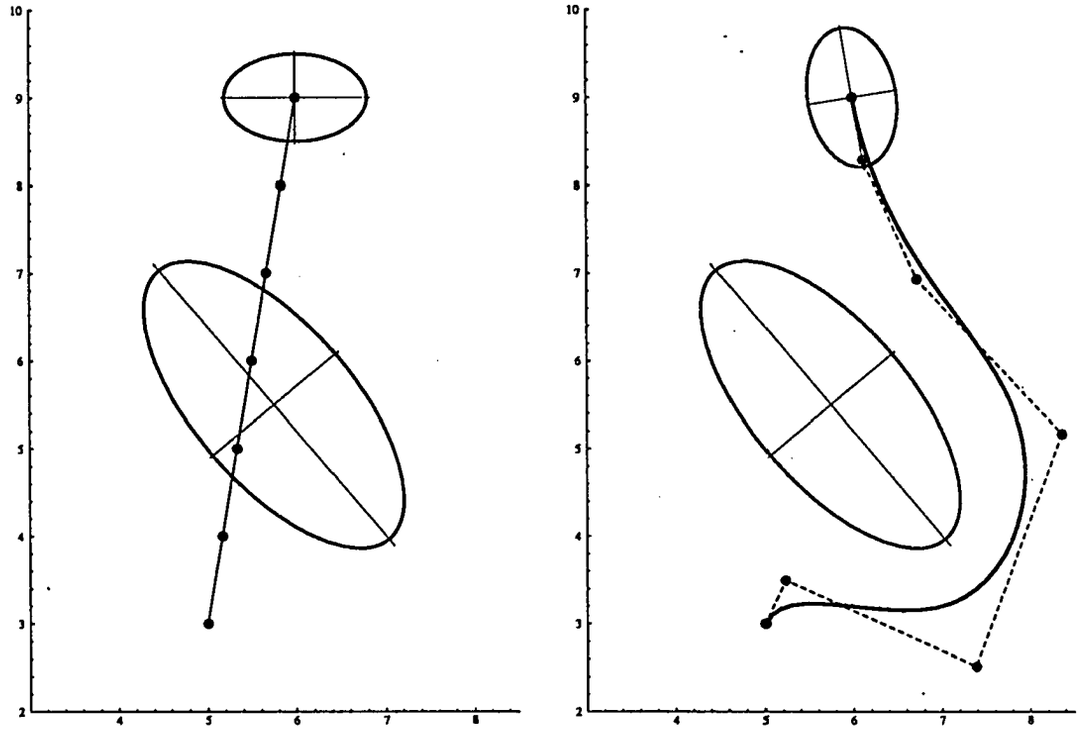
7. APPLICATION EXAMPLES

The algorithms to obtain a collision-free path for robots using simulated annealing were implemented in C on a personal DECstation 5000/25 workstation running Ultrix 4.2a and/or an IBM RISC Sytem/6000 550 running AIX 3.2. All the figures were obtained with Mathematica V2.2.

We will describe application examples with single and multiple obstacles in 2D and 3D. In Section 7.1, some 2D examples are discussed with and without a manipulator. We will start with examples in 2D. In all examples presented here, the number of curve sample points, N, was set to 50. The position of the manipulator and/or moving object is shown for every three samples.

7.1 Examples in 2D

Our first example is similar to the simple configuration shown in Figure 4.1 and will demonstrate the robustness of the simulated annealing algorithm with respect to different initial conditions as shown in Figure 7.1. For now, no link proximity cost is included. In this example, we synthesize a fourth order (i.e., a cubic) B-spline with seven control points ($n = 6$) in the vicinity of a single obstacle. The resulting knot vector, $\mathbf{U}$, is [0,0,0,0,1/4,1/2,3/4,1,1,1,1]. It is obtain as follows: The number of elements, m, is computed with Equation 3.6 yielding $m = 10$. The knot vector is



(a) Case 1 (b) Case 2

Figure 7.1: Initial configurations: (a) near global minimum and (b) near local minimum

non-periodic, then the first and last $(k+1)$ elements will be 0 and 1, respectively. In this case, we have a cubic B-spline, then the first 4 elements of $\mathbf{U}$ are equal to zero, and the last 4 elements are equal to 1. This leaves 3 knot values to determine. The knot vector should be uniform, then we should have equally spaced knots and that yields the values 1/4, 1/2, and 3/4. The cost function is calculated as described in Chapter 4 but, for now, no link proximity cost is included, i.e., $w_{in}^{L} = w_{out}^{L} = 0$.

In Figure 7.1(a), the initial position of the interior (free) control points is shown as equally distributed on the line connecting the curve end points. Since the fixed end points of the curve are skewed toward the upper left-side of the obstacle, it is clear that this simple path planning problem has two minima. One solution yields a curve which passes below the obstacle, and the other, a curve which passes over the obstacle. The global minimum, the curve with the lowest cost, is the one passing over the obstacle. Obviously, the initial configuration shown in Figure 7.1(a) is closer to the global minimum, while the configuration shown in Figure 7.1(b) is closer to the other, local, minimum.

In order to demonstrate that the simulated annealing algorithm will find the global minimum, we compared each of these initial configurations with various different initial seeds for the random number generator. The specific cost and annealing parameters for these cases were assigned as, $w_{in} = 100$, $w_{out} = 100$, $w_a = 150$, $w_p = 20$, $\epsilon_{max} = 1$. We used these values because we want to penalize more for total length of the curve and obstacle interference than for uneven distribution of the curve points. The start point, $\mathbf{P}_0$, is located at (6,9), and the goal point, $\mathbf{P}_6$, is located at (5,3). The geometric and computational results are presented in Tables 7.1 and 7.2, and a representative example solution is shown in Figure 7.2. Figure 7.3

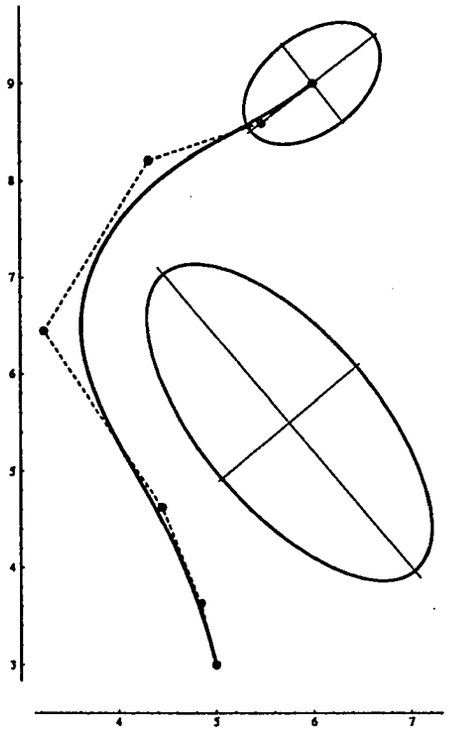


Figure 7.2: Representative solution for Case 1 and 2

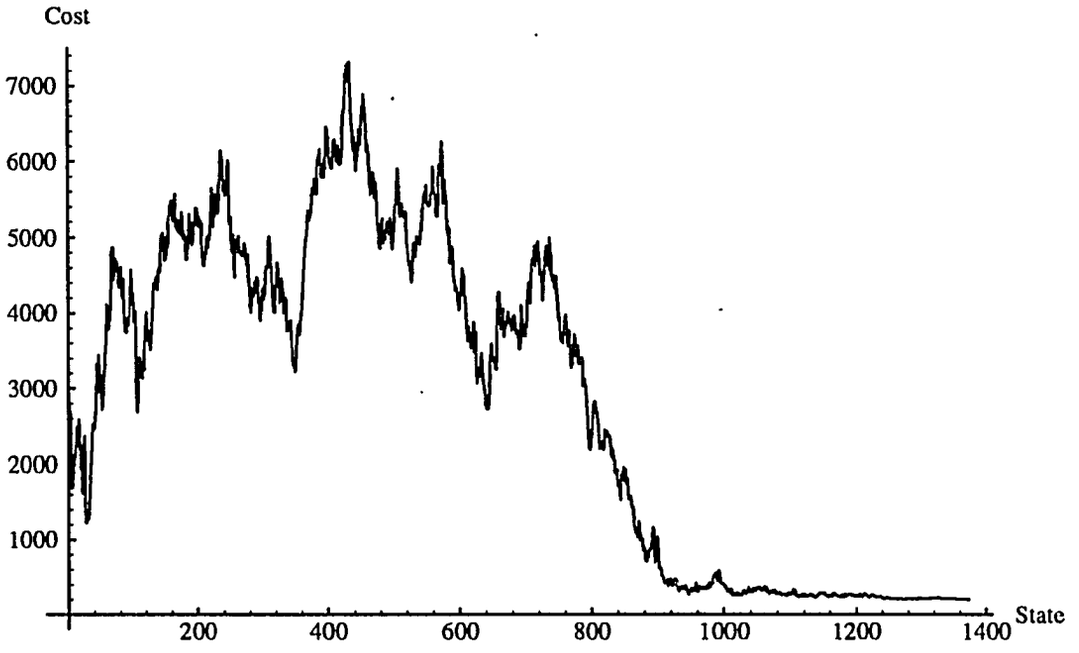


Figure 7.3: Cost function behavior of accepted states

shows the typical behavior of the cost function. Note that the simulated annealing performs a fine tuning once locating the minimum; in other words, the parametric cost starts to influence more in the cost function. One can see from the results that the algorithm reaches a solution near the global minimum with approximately the same computational effort and regardless of initial conditions.

The second example will help to demonstrate the solution of an obstacle avoidance problem with multiple objects as shown in Figure 7.4. The degree of the curve and number of control points are input parameters specified by the user. For this case, the values $k = 4$ (fourth degree B-spline) and $n = 8$ (eight control points) were selected, yielding a five-segment B-spline with non-periodic, uniform knot vector $\mathbf{U}$ = [0,0,0,0,0,1/5,2/5,3/5,4/5,1,1,1,1,1]. The other annealing parameters were:

Table 7.1: Path planning solution for Case 1

Seed	**Solution Control Points**	**Cost**	**Computation Time** (CPU seconds)
12345678	(5.59,8.60), (4.45,8.13) (3.26,6.77), (4.27,4.68) (4.72,3.67)	205.95	389.1
21436857	(5.47,8.59), (4.31,8.21) (3.23,6.44), (4.44,4.63) (4.85,3.63)	205.71	421.0
36857	(5.49,8.63), (4.26,8.09) (3.26,6.51), (4.31,4.64) (4.72,3.64)	205.57	323.5
901040	(5.63,8.74), (4.55,8.13) (3.26,6.88), (4.25,4.61) (4.66,3.57)	207.14	400.8

Table 7.2: Path planning solution for Case 2

Seed	**Solution Control Points**	**Cost**	**Computation Time** (CPU seconds)
12345678	(5.54,8.63), (4.39,8.07) (3.26,6.70), (4.33,4.70) (4.67,3.57)	203.85	316.1
542859	(5.52,8.65), (4.32,8.29) (3.26,6.45), (4.31,4.77) (4.72,3.56)	205.97	409.7
51864	(5.61,8.68), (4.28,8.19) (3.25,6.49), (4.35,4.65) (4.77,3.61)	206.31	411.4
25	(5.48,8.62), (4.30,8.08) (3.28,6.80), (4.28,4.65) (4.65,3.67)	208.37	376.9

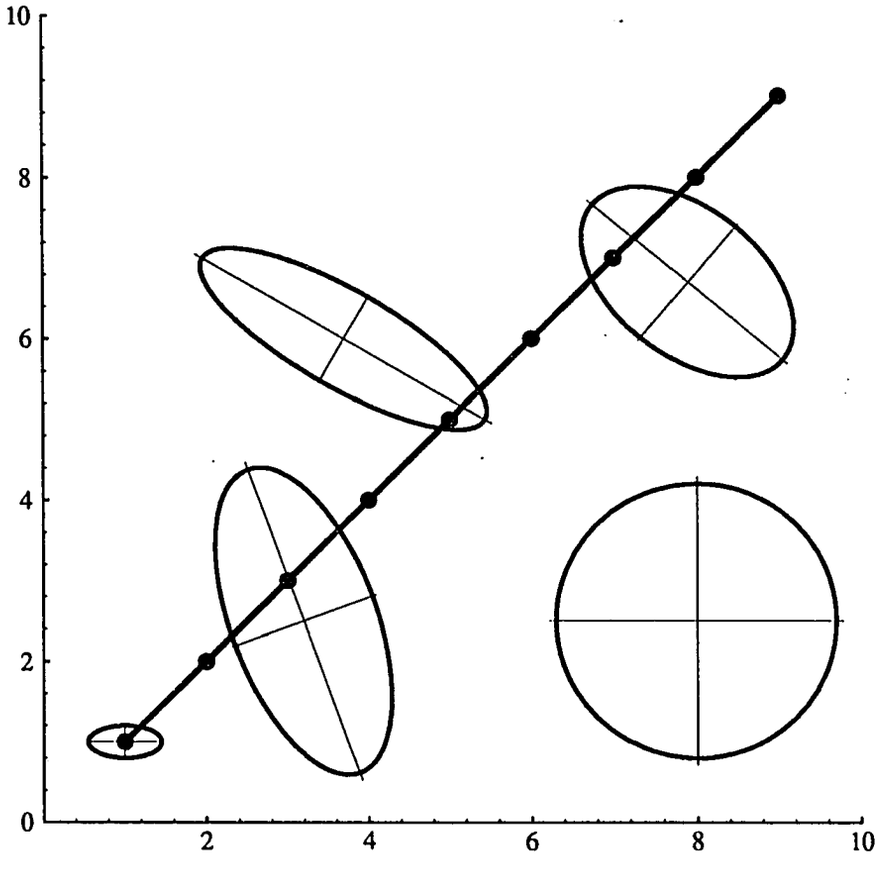


Figure 7.4: Initial configuration

$w_{in} = 150$, $w_{out} = 150$, $w_a = 300$, $w_p = 75$, $\epsilon_{max} = 1$. The start point is $\mathbf{P}_0 = (1, 1)$, and the goal points is $\mathbf{P}_8 = (9, 9)$. Again, as in the previous example, neither the computation time nor the final solution were greatly affected by the initial configuration of the control points and/or the seed for the random number generator. The CPU time for this example was $\sim$ 1000 seconds which is about twice the time taken by the previous example. This is due mainly to the increase in the number of obstacles. However, the simulated annealing algorithm proved to be robust and the minimum was very similar in each orientation case. Figures 7.5, 7.6, and 7.7 show the resulting path for each case.

In our next example, we will discuss the role weights ($w_{(\cdot)}$) play in the algorithm to obtain an optimal path when link proximity cost is included. First, we will describe an example in which a two-link planar manipulator is used to obtain the path. Figure 7.8 shows the two possible configurations of the manipulator, i.e., elbow up and elbow down, that are considered. The parameters defining the enclosing ellipse for each link is provided as input data. The link length will be taken as twice the value of the major semi-axis of its enclosing ellipse. Link 1 will be the link rotating about the origin, and link 2 will be the one following the path. This implies that the solution path should be inside the working range of the manipulator. These kinematic characteristics are handle by the manipulator link proximity cost described in Section 4.1.4. Also, due to the interference of the obstacle with link 1, the only solution for either configuration shown in Figure 7.8 is that link 1 must rotate in the counterclockwise direction to reach the goal point.

The selected degree, k, and number of control points, n, were 3 and 6, respectively. Here, in order to reach the goal point, we reduce the weights of the cost due to

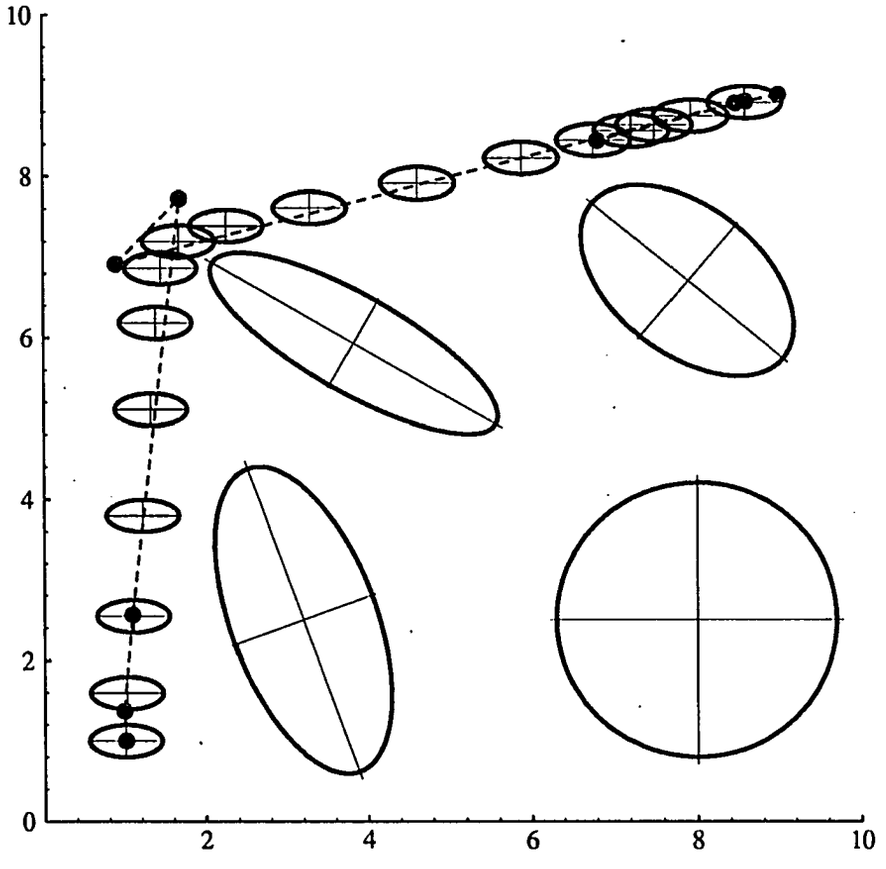


Figure 7.5: Path planning problem with multiple obstacles and fixed orientation for moving object, $\theta_M = 0°$

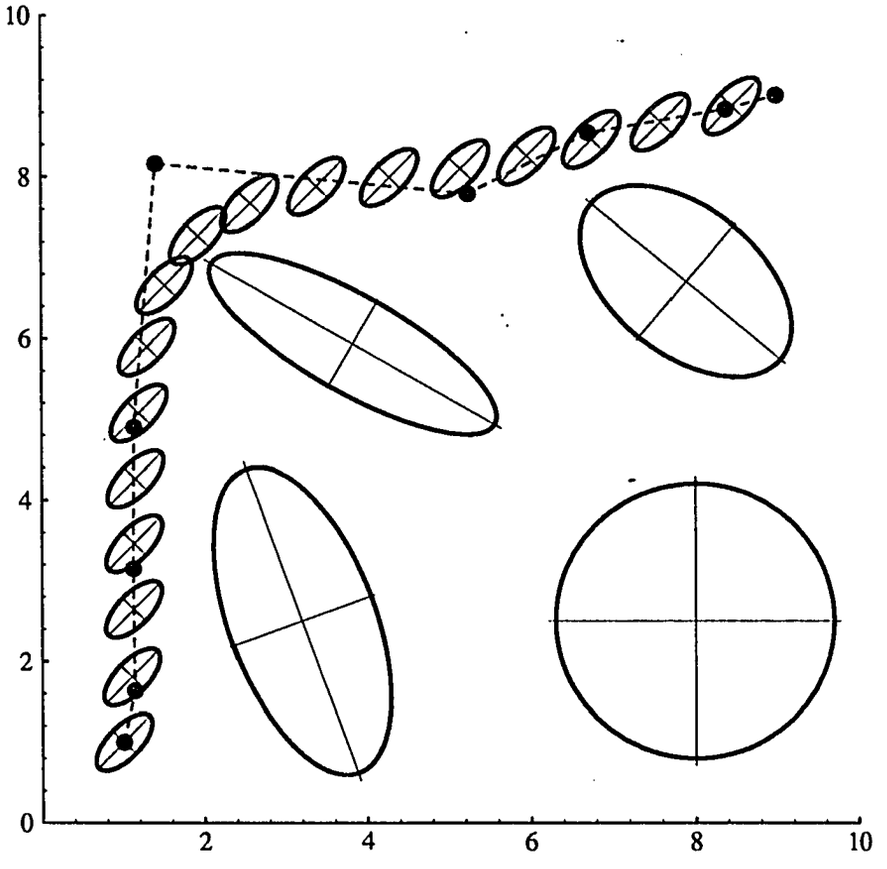


Figure 7.6: Path planning problem with multiple obstacles and orientation of moving object as another d.o.f., $\theta_M = 45.6°$

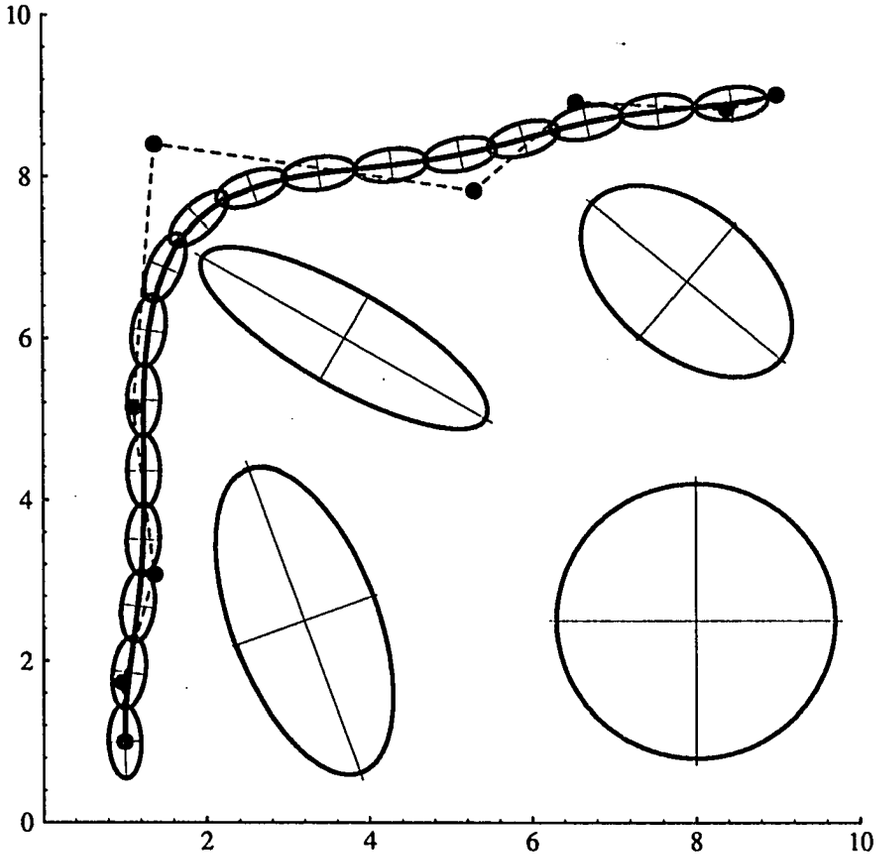


Figure 7.7: Path planning problem with multiple obstacles and variable orientation for moving object

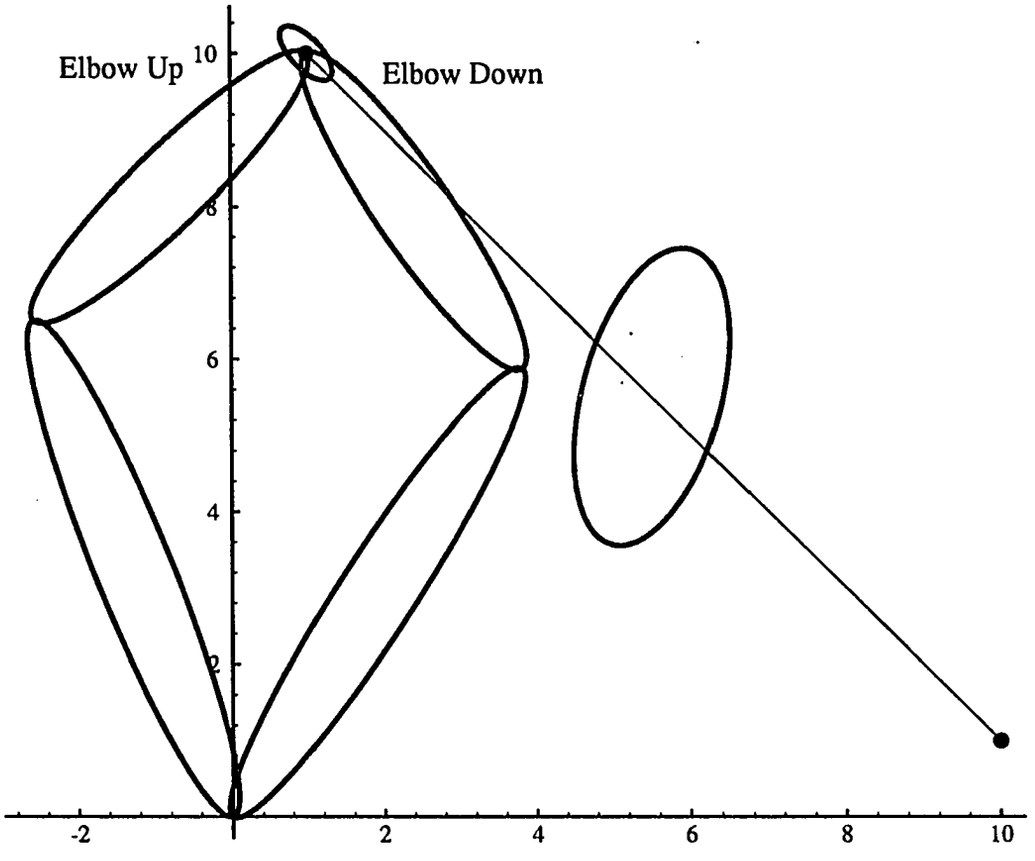


Figure 7.8: Path planning problem when kinematic characteristics of a manipulator are included

excessive arc length and uneven parametric distribution. These weights were in the order of 1/20 in relation to the other weights. These are the values used: $w_{in} = 10$, $w_{out} = 10$, $w_a = 0.5$, $w_p = 0.5$, $w_{in}^L = 10$, $w_{out}^L = 10$, and $\epsilon_{max} = 1$. The start point, $\mathbf{P}_0$, is located at (1,10) and the goal point, $\mathbf{P}_6$ is at (10,1). As in the previous example, the initial location of the interior B-spline control points is in a straight line from $\mathbf{P}_0$ to $\mathbf{P}_6$. The solution to the obstacle avoidance problem is shown in Figure 7.9 and was found in about 880 seconds with a final cost of 6.86. Figure 7.10 shows the solution path and its control point polygon. The solution shown in Figure 7.9 was reached by testing several weight values for each configuration of the manipulator shown in Figure 7.8. The program developed for this research work is sufficiently flexible to incorporate bounds on the joint angles of the manipulator. This feature can help to obtain the solution path for the obstacle avoidance problem. In this case, we did not allow the rotation angle of link 1 to take values in the range 25° to 55°.

Now, the importance of each weight can be seen in Figure 7.11 and 7.12. Figure 7.11 shows a "solution" when the parametric weight is too high compared to the others. Here, all the weights have the same value, i.e. $w_{in} = w_{out} = w_a = w_p = w_{in}^L = w_{out}^L = 10$. This will force the algorithm to come up with a solution in which either the points are evenly distributed along a path in which there exists interference between link 1 and the obstacle, see Figure 7.11, or most of the points are located in either upper-left or lower-right side of the obstacle and there also exists interference, see Figure 7.12. This is the main reason why one should check the resulting solution to look for any interference.

Since the algorithm will only consider the numerical minimization of the cost function, one should provide additional information to determine, during the mini-

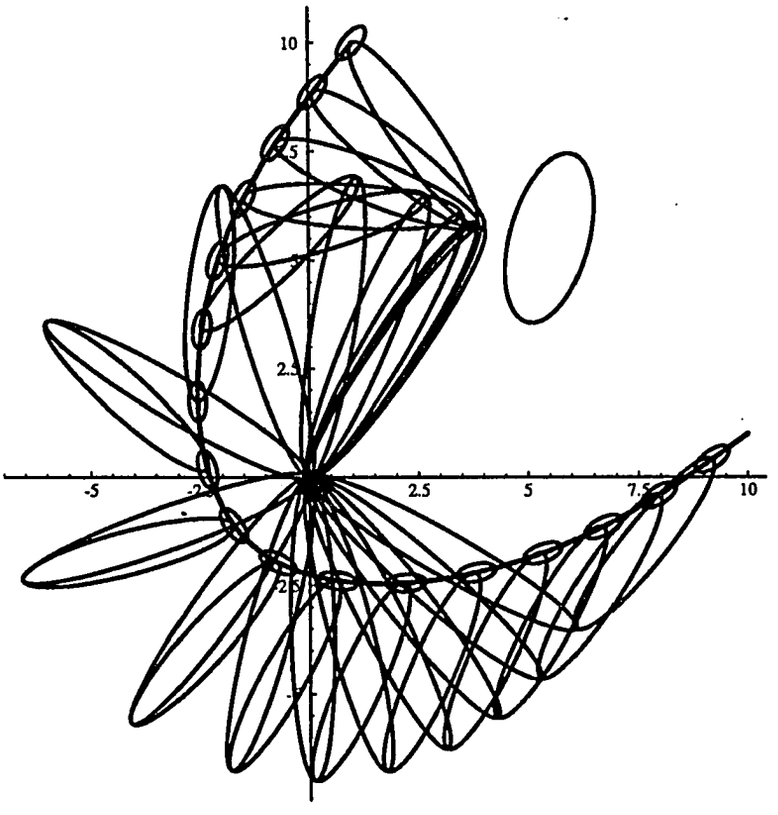


Figure 7.9: . Solution to an obstacle avoidance problem when link proximity cost and bounds on joint angles are included

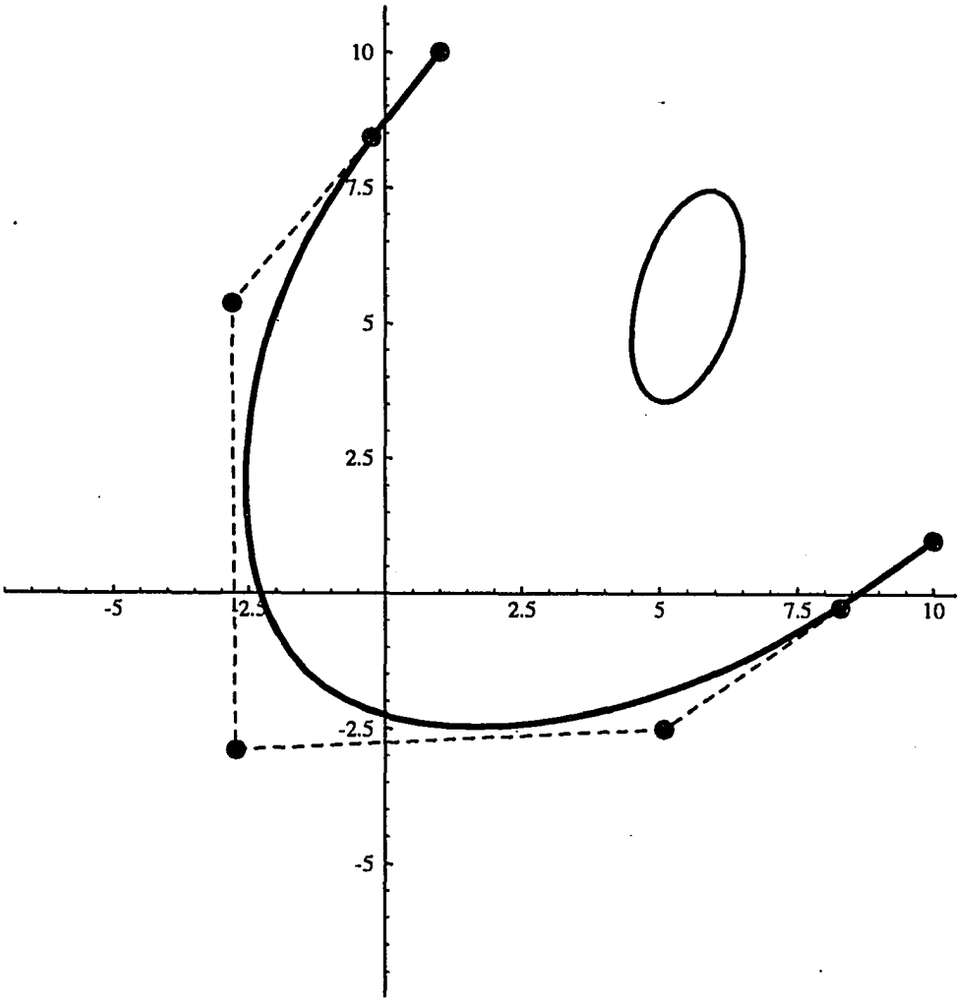


Figure 7.10: Solution path and control point polygon for Figure 7.9

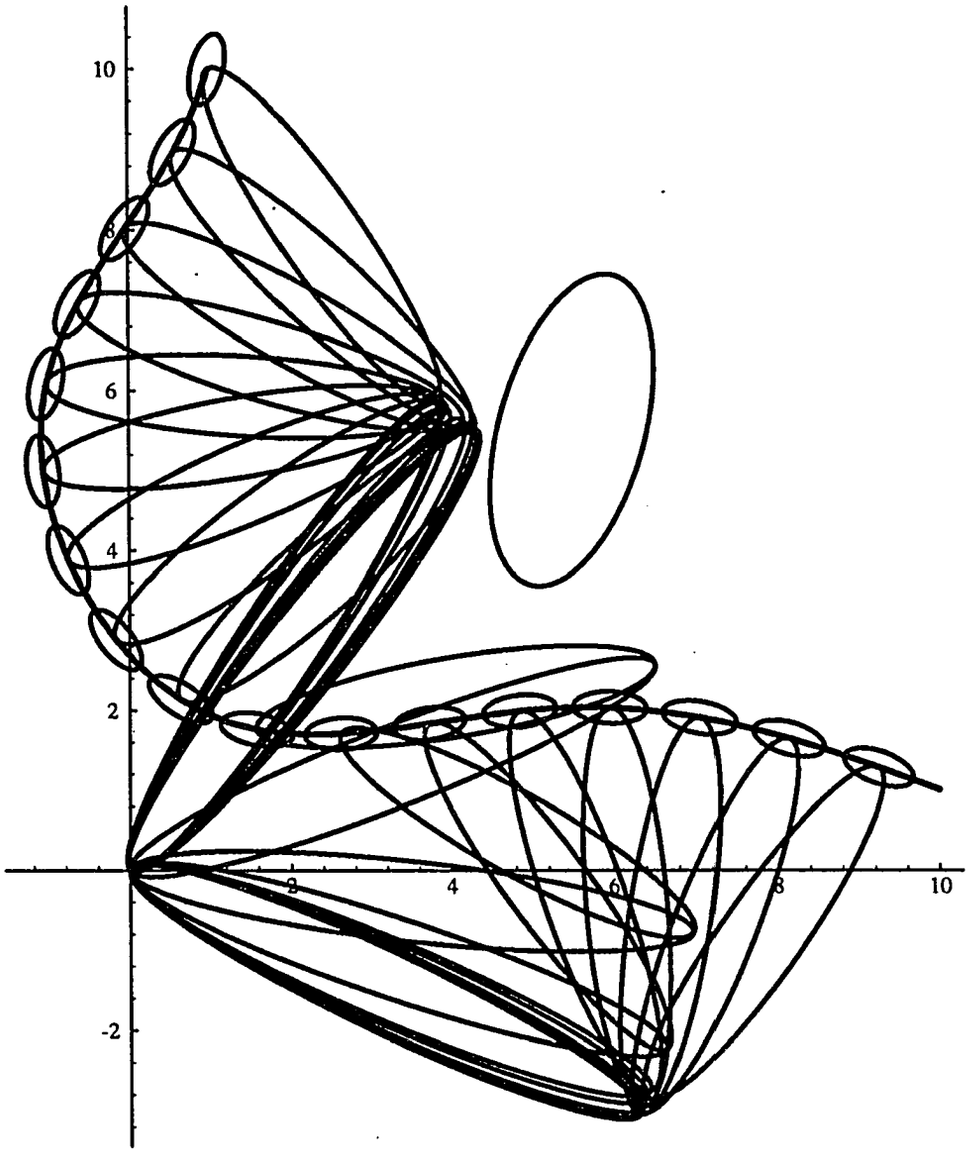


Figure 7.11: Solution with good parametric distribution; however, there exists interference

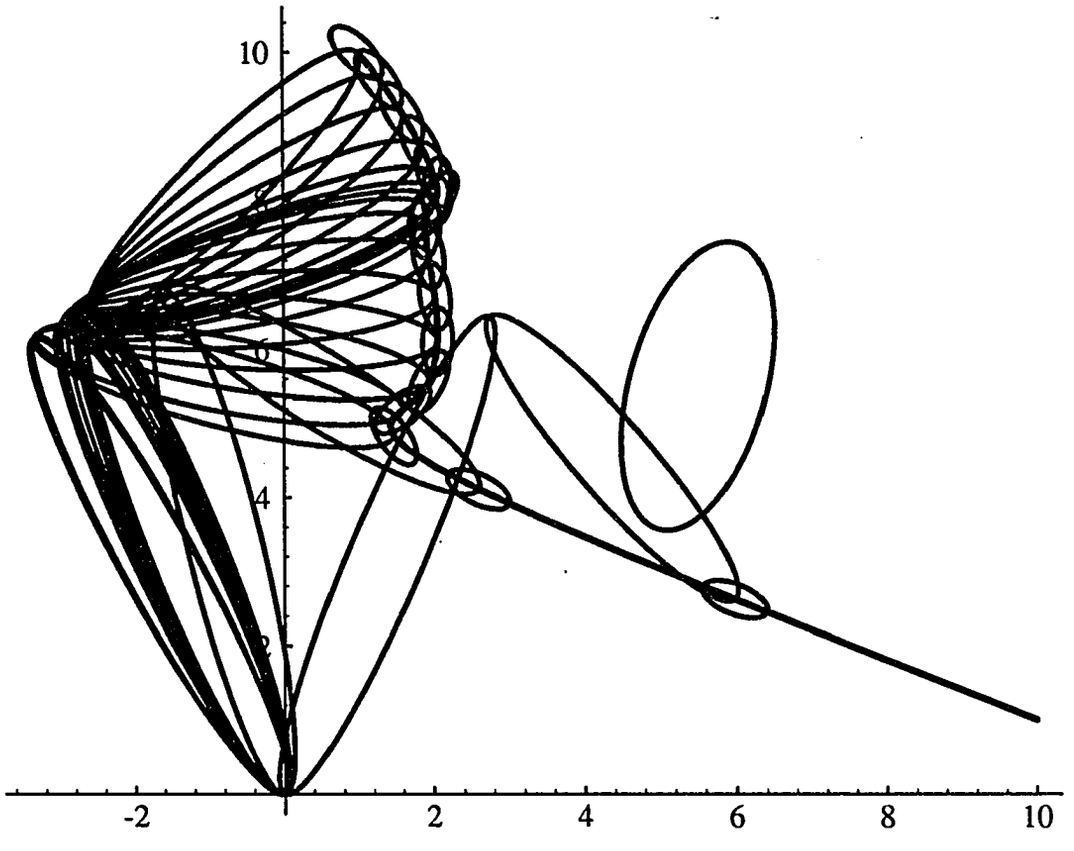


Figure 7.12: Solution when most of the points are on one side of the path

mization process, whether or not the path is a valid one, and then incorporate that into the program developed for this purpose. This is not an easy task mainly because it is highly dependent on the application of this technique. However, we can show the most important points to be considered and that way we apply this technique to a wide variety of collision-free path planning problems.

The next example is very similar to the previous one. However, in this case there is no possible interference between link 1 and the obstacle. Figure 7.13 shows the initial configuration used, the final path and its control point polygon obtained with the proposed technique. The parameters used were: $k = 3$, $n = 6$, the cost weights were all equal to 5, $\epsilon_{max} = 1$, $\mathbf{P}_0 = (6, 9)$, and $\mathbf{P}_6 = (5, 3)$. The solution, shown in Figure 7.14, was reached in 634.6 seconds with a final cost equal to 39.97. Figure 7.15 shows the behavior of the cost function for this obstacle avoidance problem.

Figure 7.16 shows another solution when the weight for the parametric cost is high and the weight for the arc length is set low. Figure 7.17 shows the final path and its control point polygon. This relation between the two weights gives a good parametric distribution but produces a lengthy path. Care must be taken when choosing these two weights.

7.2 Examples in 3D

We will start with a simple 3D path planning problem in which a single obstacle is present. The obstacle is located at $(5, 5; 5)$ with Euler angles $\psi = -22°$, $\theta = -49°$, and $\phi = -25°$. The semi-axes values are $a = 2$, $b = 1.3$, and $c = 0.8$ with a clearance value of 0.2. The dimensions of the moving object semi-axes are $a = 0.8$, $b = 0.5$, and $c = 0.35$. The orientation of the moving object is given by the first derivative of

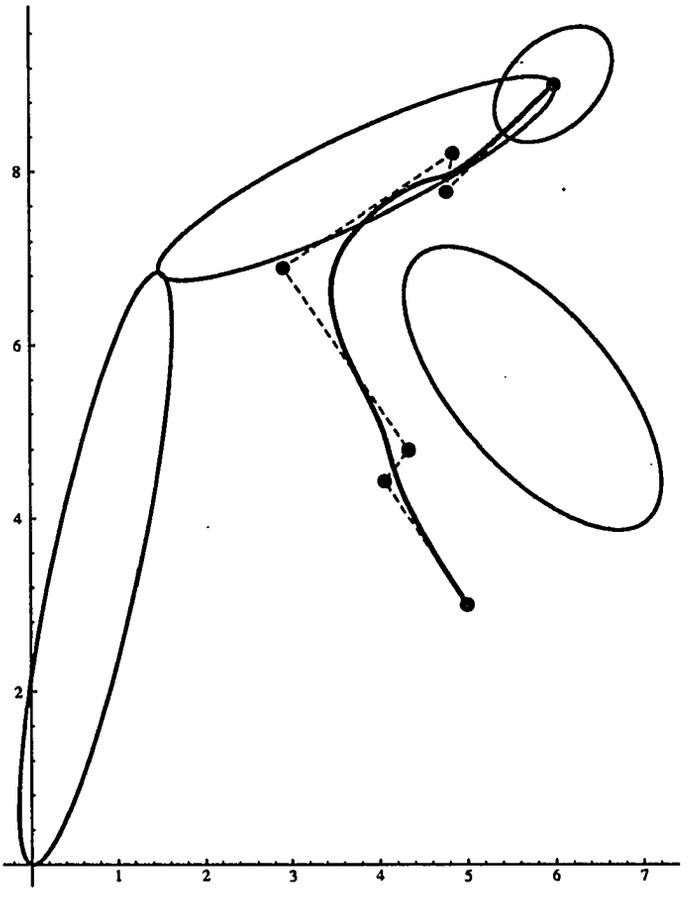


Figure 7.13: Initial configuration for the manipulator and final path with its control point polygon

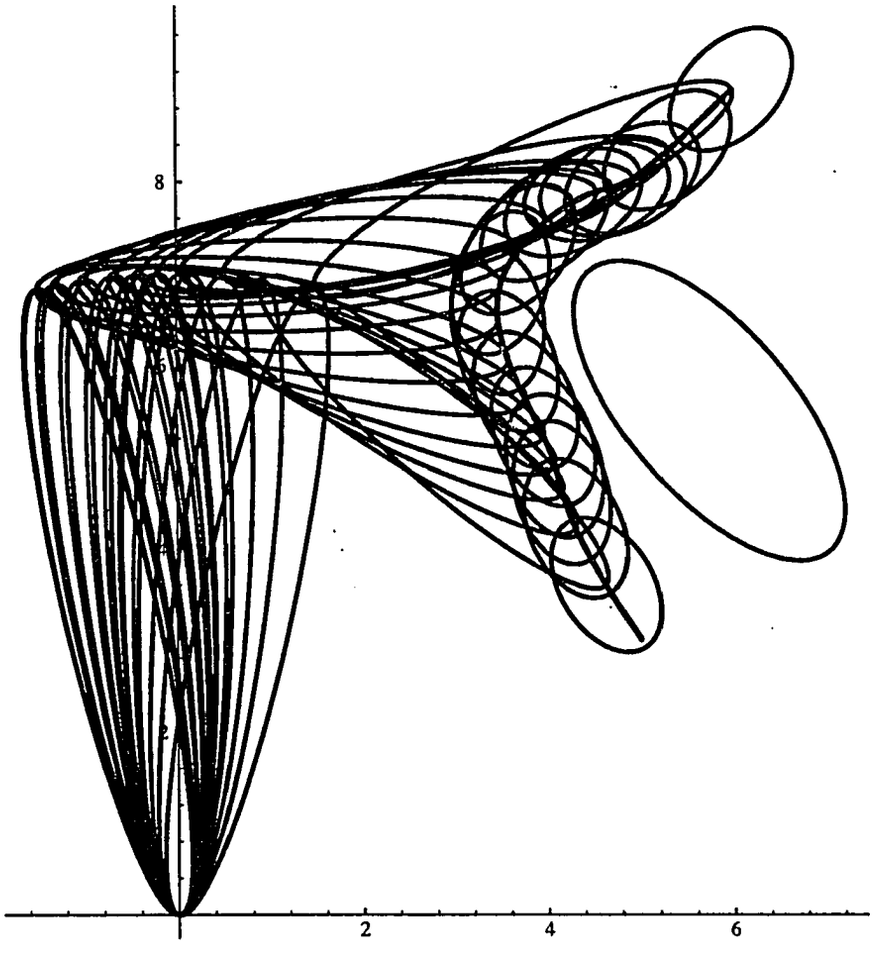


Figure 7.14: Solution to a path planning problem with no possible interference bewtween the obstacle and link 1 of the manipulator

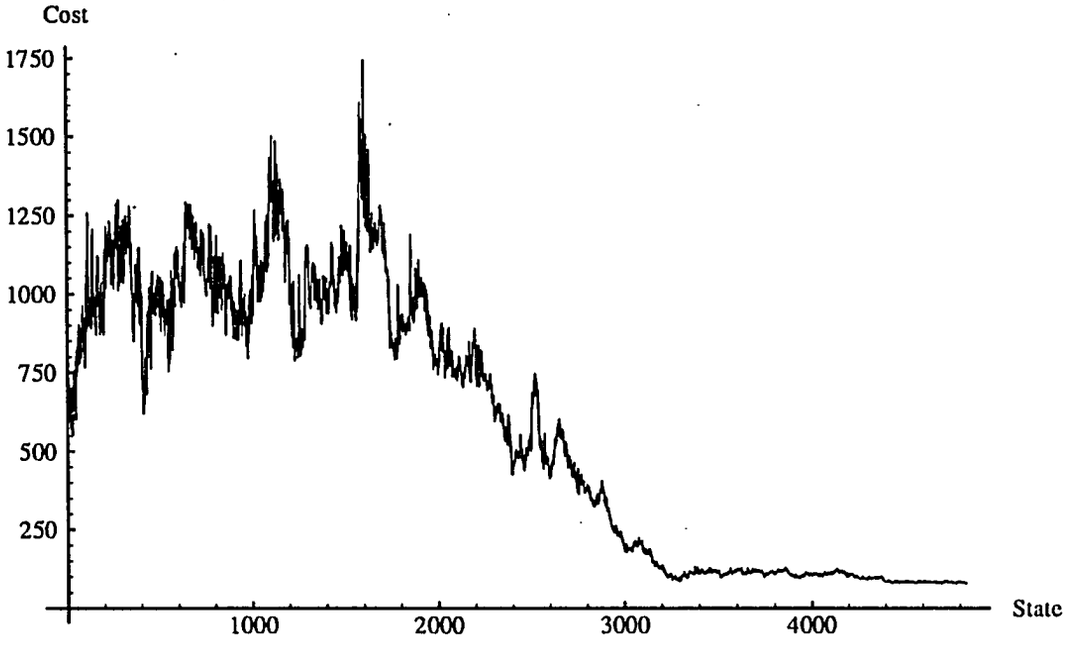


Figure 7.15: Behavior of the cost function for path shown in Figure 7.14

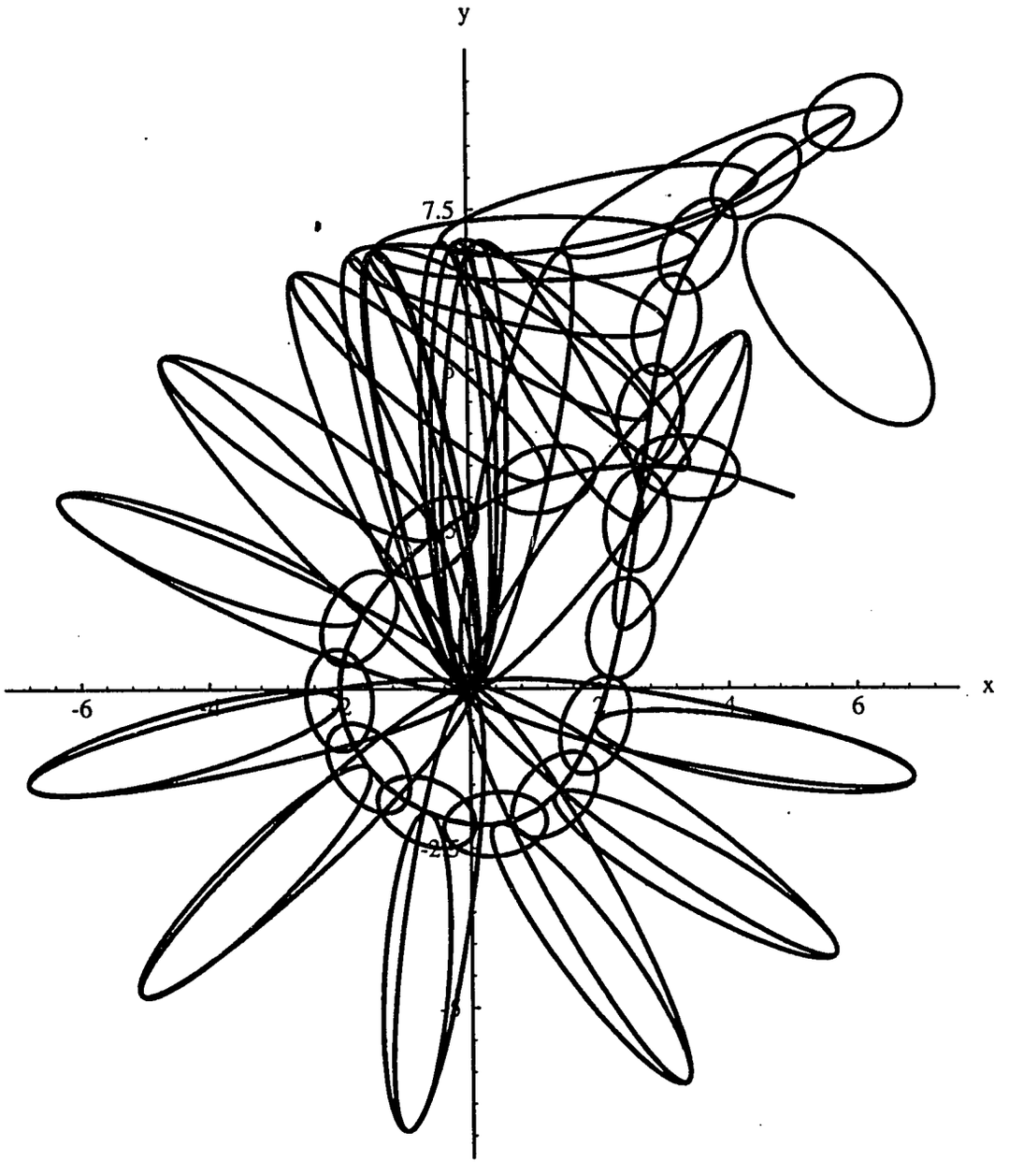


Figure 7.16: Solution to the path planning problem with w_p high and w_a low

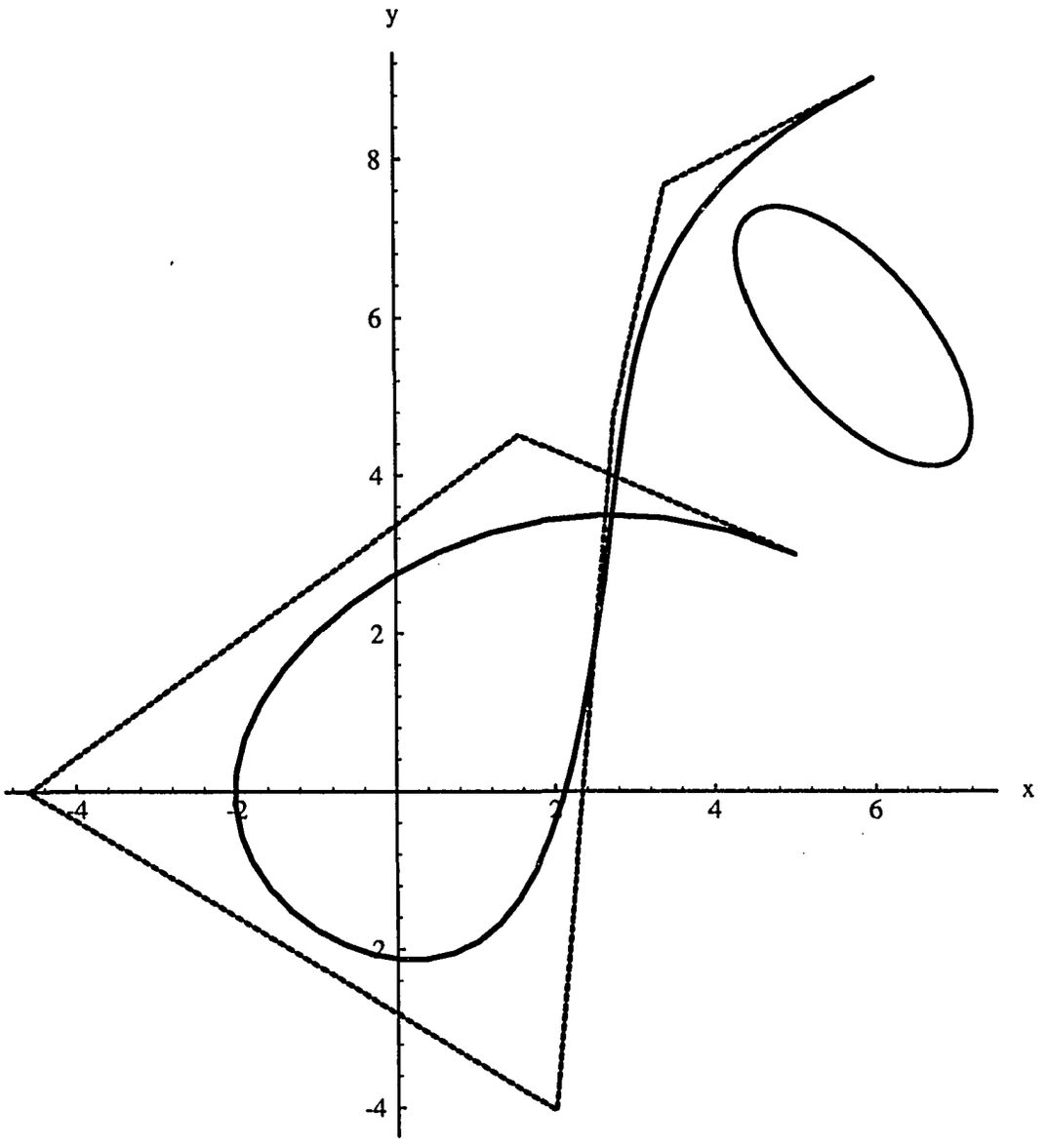
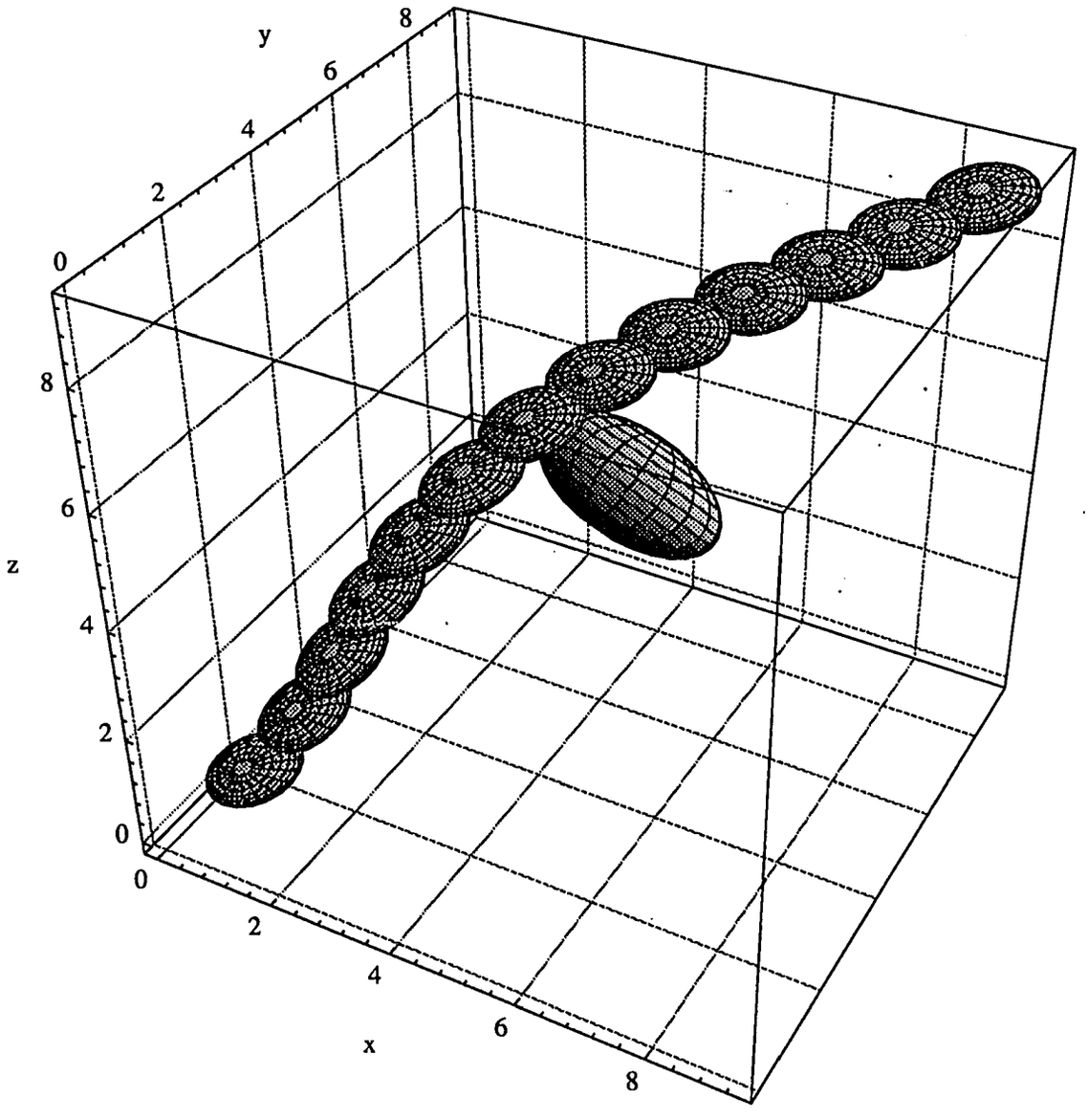


Figure 7.17: Path and control point polygon with w_p high and w_a low

the B-spline curve. The degree of the B-spline is $k = 3$ with $n = 6$ control points. The start and goal points are $\mathbf{P_0} = (1, 1, 1)$ and $\mathbf{P_6} = (9, 9, 9)$, and the other parameters are $w_{in} = 10$, $w_{out} = 10$, $w_a = 25$, and $w_p = 10$. No kinematic characteristic of a manipulator were included for this example. Figure 7.18 shows the solution which the simulated annealing algorithm reached in ~ 3000 seconds.

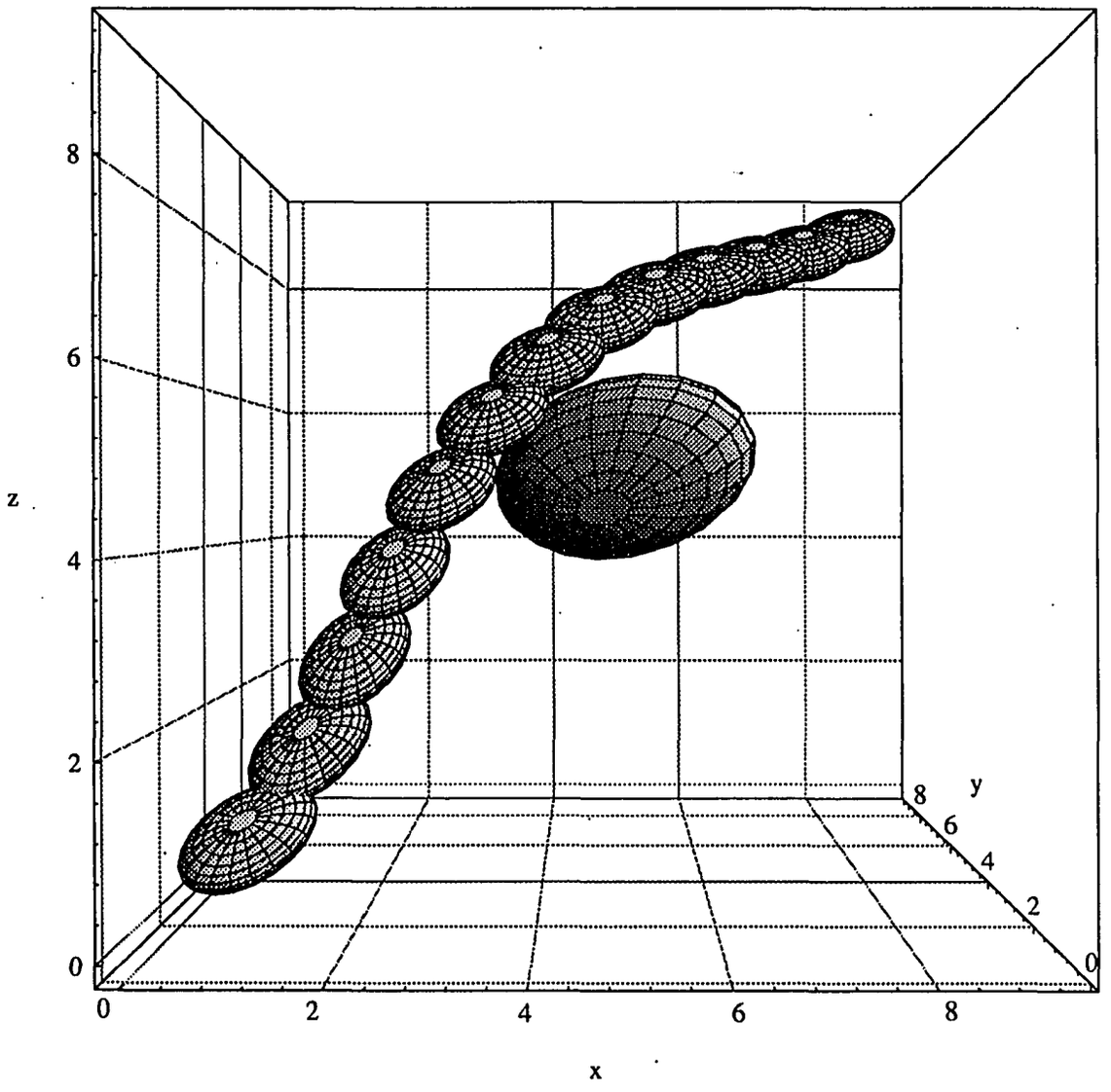
Our last example is with multiple obstacles. A fourth degree B-spline was chosen with 8 control points. The other parameters were $w_{in} = 350$, $w_{out} = 350$, $w_a = 450$, $w_p = 100$, and $\epsilon_{max} = 1$. No link proximity cost was included. The semi-axes dimensions, Euler angles, and location of each obstacle and moving object are given in Table 7.3. As in the previous example, the orientation of the moving object is given by the first derivative of the B-spline curve.

Figure 7.19 shows the solution to the example, and it was reached in about 3700 seconds. Note that the increase in the number of obstacles increased the computation time by less than a third. We cannot give a minimum time in which the simulated annealing algorithm would reach a solution.



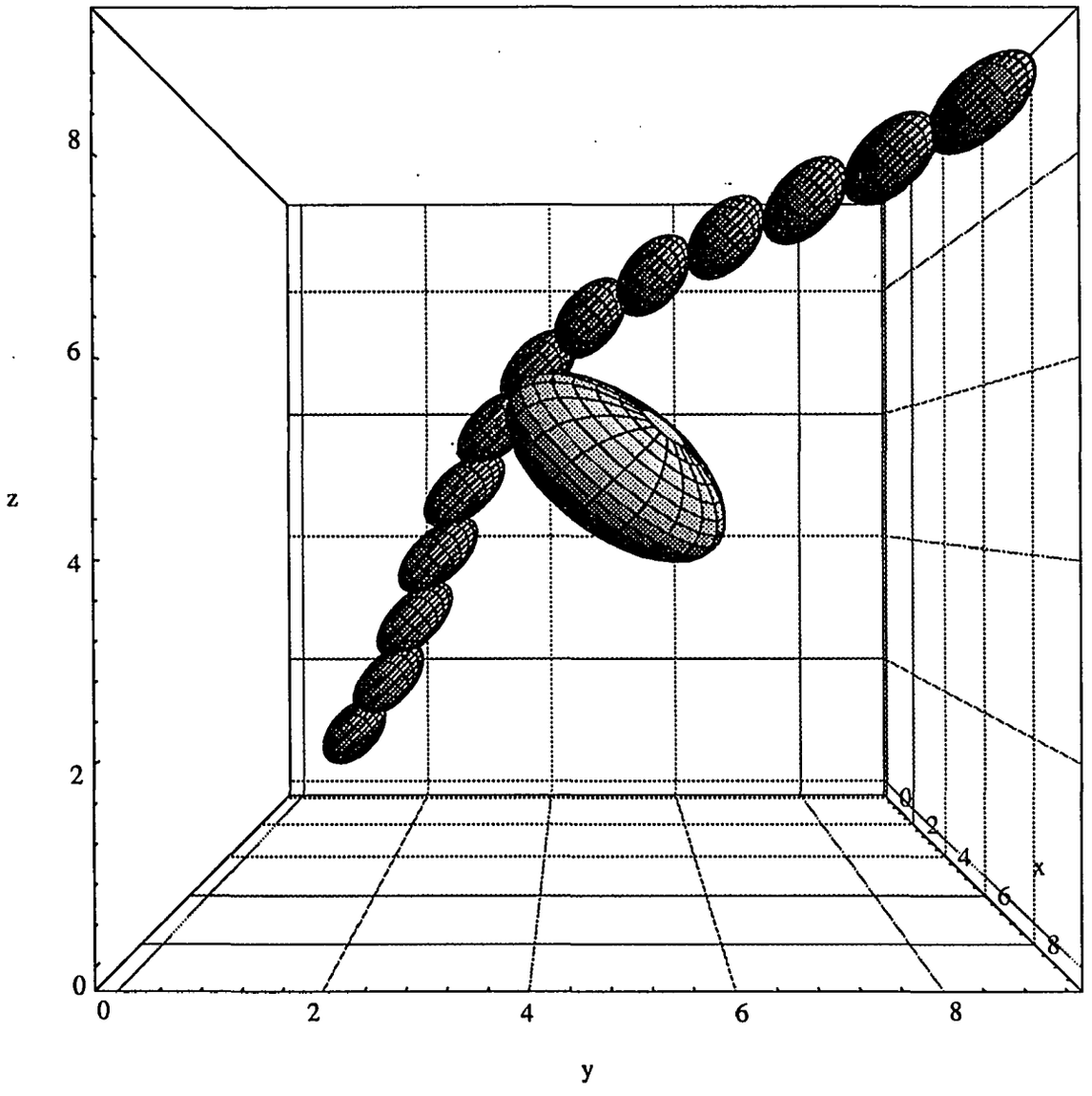
(a) Perspective

Figure 7.18: A simple 3D path planning problem



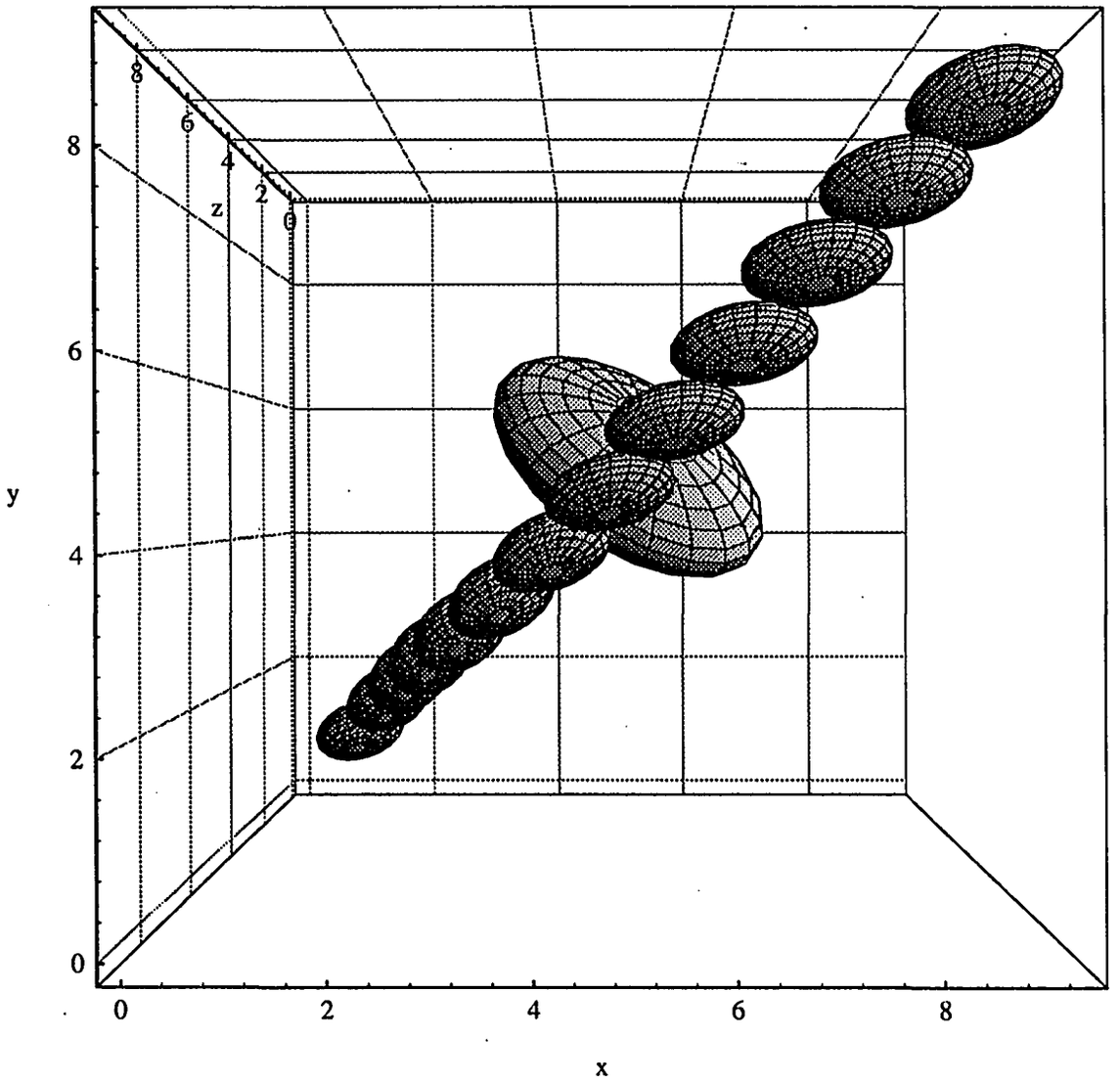
(b) Front

Figure 7.18 (Continued)



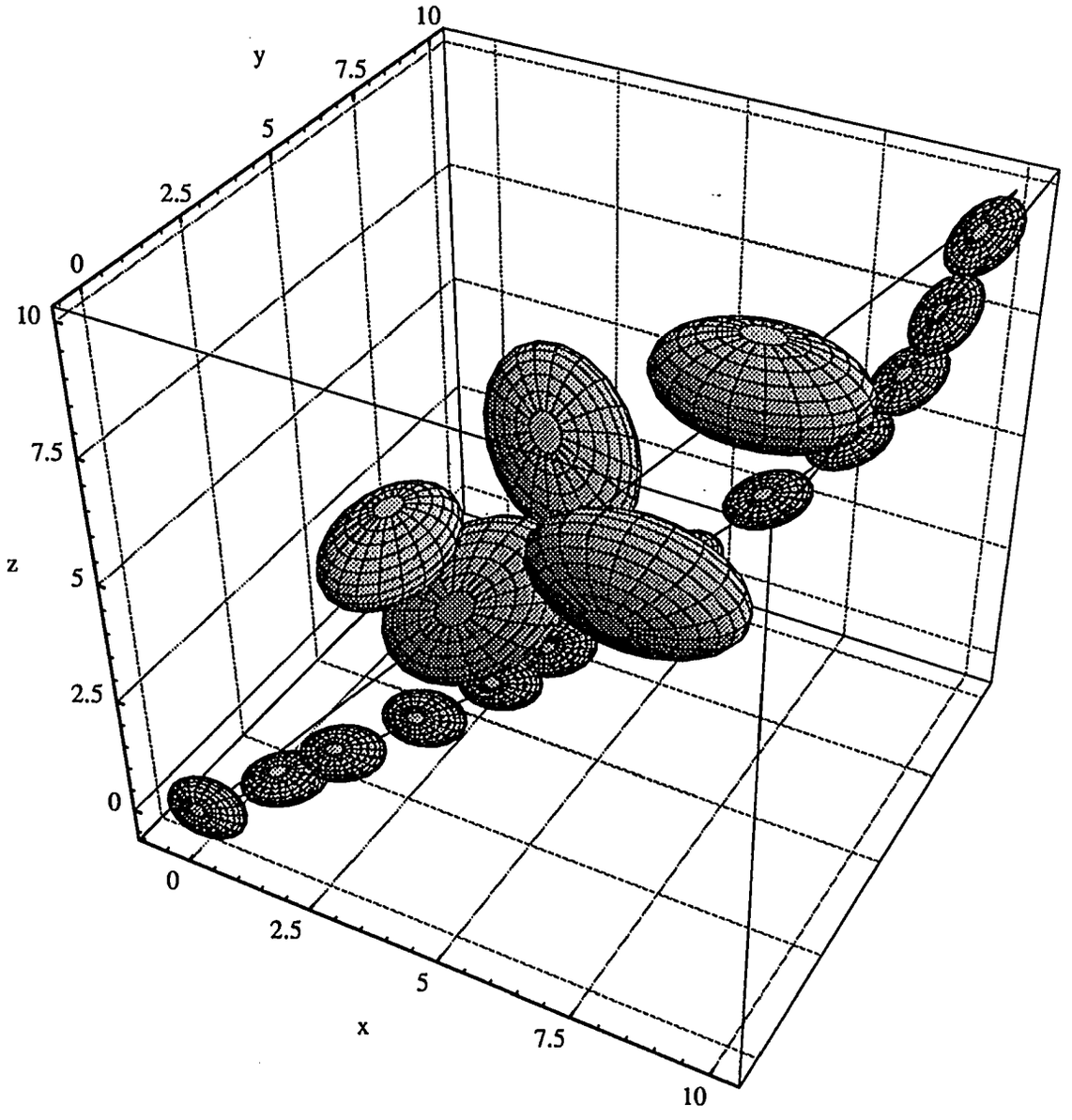
(c) Right side

Figure 7.18 (Continued)



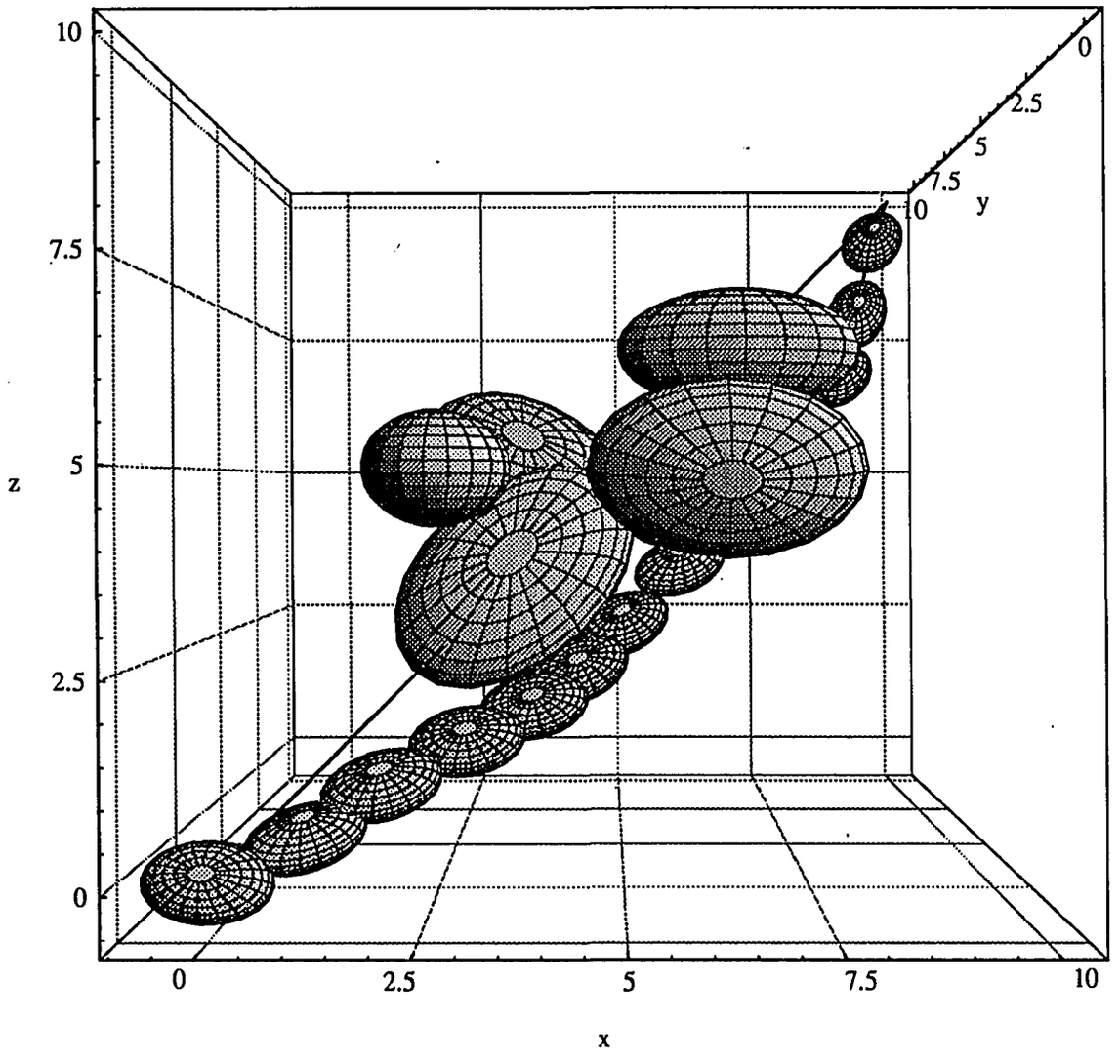
(d) Above

Figure 7.18 (Continued)



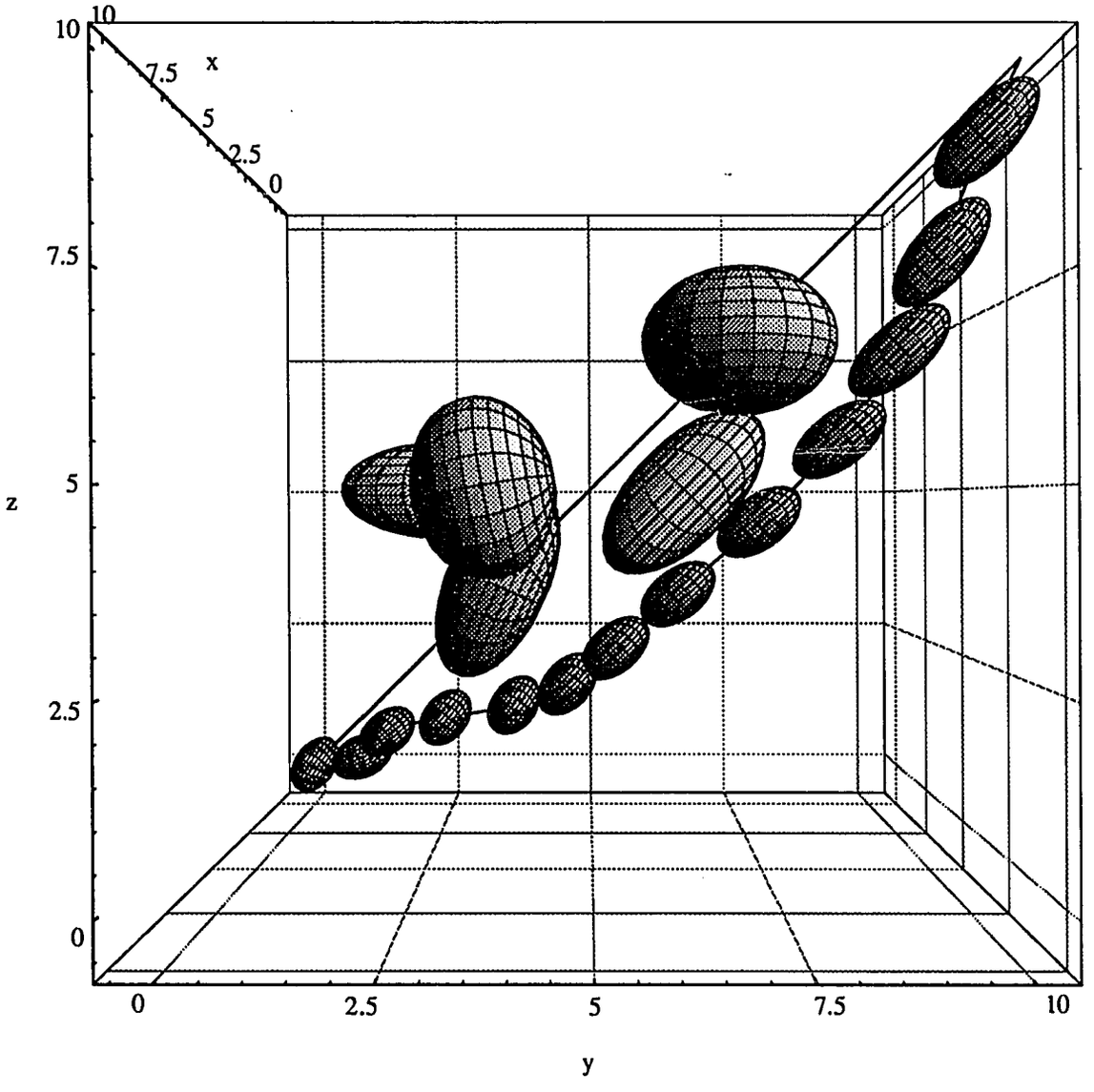
(a) Perspective

Figure 7.19: Path planning solution with multiple obstacles in 3D



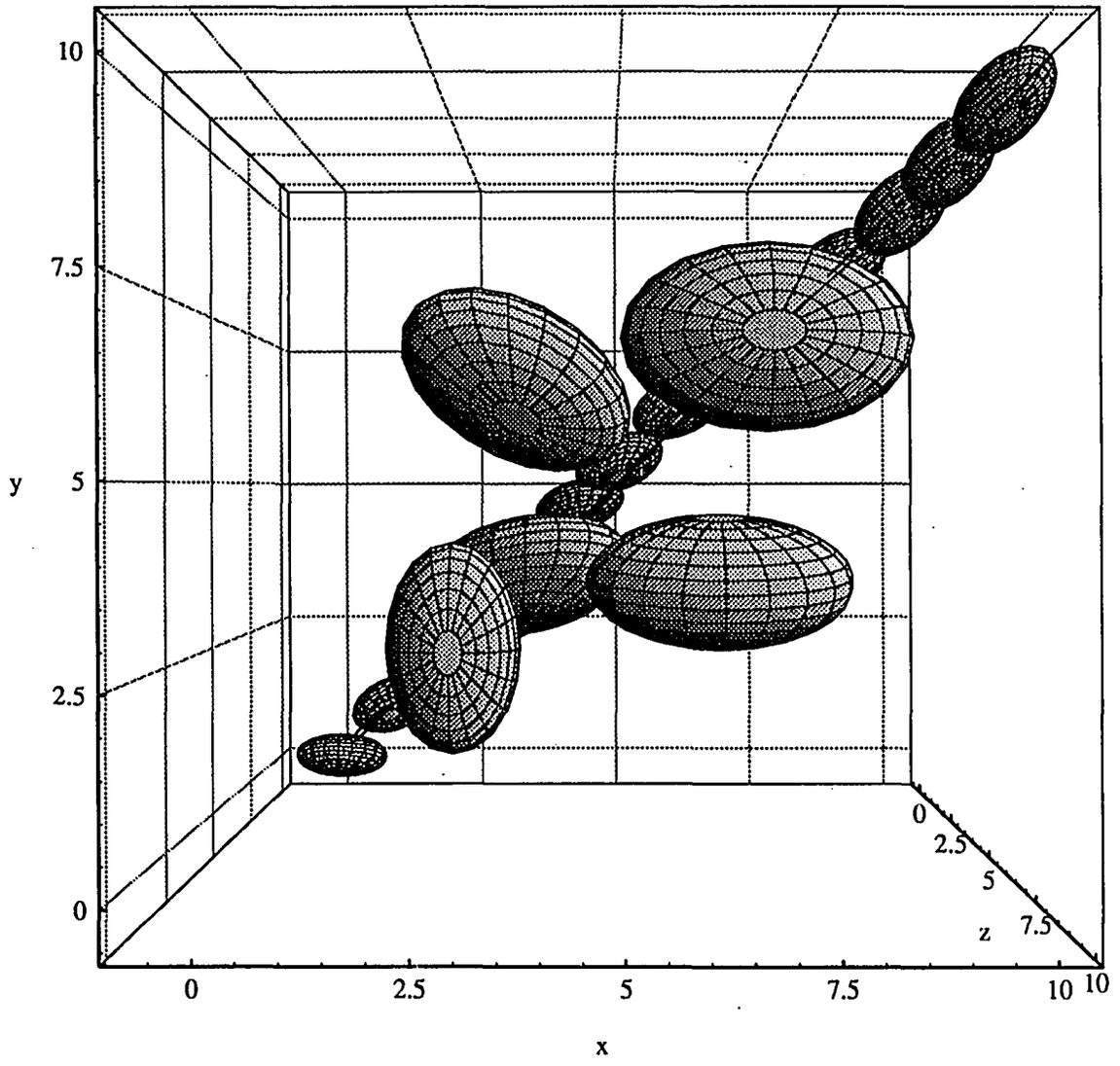
(b) Front

Figure 7.19 (Continued)



(c) Right side

Figure 7.19 (Continued)



(d) Above

Figure 7.19 (Continued)

Table 7.3: Ellipsoid parameter values for Figure 7.19

Ellipsoid	**Semi-axes**	**Euler angles**			**Location**
	(a, b, c)	$(\psi$	θ	$\phi)^\circ$	(x, y, z)
1	2.0, 1.3, 0.80	0	65	45	(3.5, 3.5, 3.5)
2	1.6, 1.0, 0.80	90	0	0	(2.5, 2.5, 3.5)
3	2.0, 1.3, 0.80	0	-45	-45	(3.5, 6.5, 5.0)
4	2.0, 1.3, 1.00	0	-80	0	(6.5, 3.5, 5.0)
5	2.0, 1.3, 1.00	0	0	0	(7.0, 7.0, 7.0)
Moving	0.8, 0.5, 0.35	Not applicable			

8. CONCLUSIONS

This thesis presents a technique to plan collision-free paths for robots in 2 and 3 dimensions using B-splines and simulated annealing. Uniform, non-periodic, non-rational B-spline curves are synthesized to solve the path planning problem. The simulated annealing algorithm is used as the optimization technique, and the design variables are the coordinates of the B-spline control points.

Objects are modeled with ellipsoid type shapes. The parameters of the ellipsoids are obtained by formulating a minimization problem in which the area and the volume of the ellipsoid are used as objective functions for two and three dimensions, respectively. An optional objective function, using minimization of the approximating error, is also discussed. These minimization problems are constrained so that the resulting ellipsoid must enclose the object. The area/volume cost function showed to be more robust in yielding good ellipsoid model parameters.

A cost function with object proximity, arc length, parametric, and, possibly, link proximity cost components was developed for the simulated annealing algorithm. Since the purpose of this work is path planning and not position synthesis, a simple two-link planar manipualtor synthesis is described to show that characteristics of robots, which will perform the task, can be included in this technique to solve the path planning problem.

An optimization problem is formulated to obtain the minimum distance between two ellipsoids. The problem is solved by finding stationary points of the resulting augmented optimization problem. A globally convergent Newton's method is used to accomplish this task. In very rare situations, this algorithm fails to converge to a solution due to poor initial conditions. This can be solved by trying another intial solution. The solution of this optimization problem also helps to determine if there exists interference between the two ellipsoids.

The simulated annealing algorithm with its minor modifications proved to be robust in finding an optimal solution for our collision-free path planning problem. This solution depends on the values of the cost component weights, i.e., w_{in}, w_{out}, w_a, w_p, etc. and the other input parameters. Therefore, several tests with different seeds for the random number generator, and different parameter values should be carried out before accepting a solution as a "good solution".

The present work has shown a technique to obtain a continuous collision-free path for an AGV and/or robot. The moving object is not just considered as a point as in [14, 34, 63, 98, 99, 105, 140, 186] but as an actual object and therefore, its orientation is considered to obtain the path. This is one the main characteristics of the work. Also, most of the work in path planning to date is in two dimensions [14, 17, 25, 34, 43, 52, 59, 63, 98-101]. The present technique can be easily applied for 2D and 3D curve synthesis.

Many avenues for future work exist. Rational B-splines give an additional capability to locally modify (by knot segments) the curve which might yield a better (lower cost) path. More characteristics of the robot dynamics could be included to get joint time history and a better path for specific applications, and extend them

to 3D. Look at robot arm for interference as a separate issue. A technique to choose some good default input parameter values, i.e., degree of the B-spline, number of control points, initial temperature, weights, percentage rate of accepted states, and number of outer loop iterations, would save time usually spent on additional tests. Alternate distribution of the weight values so the computer could do this part of the task.

BIBLIOGRAPHY

[1] D. Abramson. A very high speed architecture for simulated annealing. *Computer*, 25:27–36, May 1992.

[2] Ronald C. Arkin. Homeostatic control for a mobile robot: dynamic replanning in hazardous environments. *Journal of Robotic Systems*, 9:197–214, March 1992.

[3] J. Barraquand and J.-C. Latombe. Robot motion planning: a distributed representation approach. *International Journal of Robotics Research*, 10:628–49, December 1991.

[4] J. Beardwood, J.H. Halton, and J.M. Hammersley. The shortest path through many points. *Proc. Cambridge Philos. Soc.*, 55:299–327, 1959.

[5] Z. Bien and J. Lee. A minimum-time trajectory planning method for two robots. *IEEE Transactions on Robotics and Automation*, 8:414–18, June 1992.

[6] K. Binder. *Monte Carlo Methods in Statistical Physics.* Springer, New York, 1978.

[7] J.E. Bobrow. Optimal robot path planning using the minimum-time criterion. *IEEE J. Rob. and Autom.*, 4(4):443–450, August 1988.

[8] M. Boehm. Inserting new knots into b-spline curves. *Computed-Aided Design*, 12(04):199–201, May 1980.

[9] M. Boehm and H. Prautzsch. The insertion algorithm. *Computed-Aided Design*, 17(02):58–59, March 1985.

[10] I. Bohachevsky, M.E. Johnson, and M.L. Stein. Generalized simulated annealing for function optimization. *Technometrics*, 28:209–217, 1986.

[11] W. Böhm. Efficient evaluation of splines. *Computing*, 33:171–177, 1984.

[12] J. Borenstein and Y. Koren. The vector field histogram–fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7:278–88, June 1991.

[13] R. A. Brooks. Planning collision-free motions for pick-and-place operations. *Int. J. of Robotics Research*, 2(4):19–44, 1983a.

[14] R. A. Brooks and T. Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Trans. on Systems, Man and Cybernetics*, SCM-15(2):224–233, 1985.

[15] R.E. Burkard and F. Rendl. A thermodynamically motivated simulation procedure for combinatorial optimization problems. *European J. of Oper. Res.*, 17:169–174, 1984.

[16] C. R. Carignan. Trajectory optimization for kinematically redundant arms. *Journal of Robotic Systems*, 8:221–48, April 1991.

[17] W. F. Carriker, P. K. Khosla, and B. H. Krogh. Path planning for mobile manipulators for multiple task execution. *IEEE Transactions on Robotics and Automation*, 7:403–8, June 1991.

[18] A. Casotto, F. Romeo, and A.L. Sangiovanni-Vincentelli. A parallel simulated annealing algorithm for the placement of macro-cells. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, pages 30–33, Santa Clara, November 1986.

[19] A. S. Cela and Y. Hamam. Optimal motion planning of a multiple-robot system based on decomposition coordination. *IEEE Transactions on Robotics and Automation*, 8:585–96, October 1992.

[20] J. Cesarone and K.F. Eman. Efficient manipulator collision avoidance by dynamic programming. *Robotics and Computer-Integrated Manufacturing*, 8(1):35–44, 1991.

[21] Y.C. Chen and M. Vidyasagar. Optimal trajectory planning for a planar n-link manipulators in the presence of obstacles. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 202–208, 1988.

[22] R. M. H. Cheng and R. Rajagopalan. Kinematics of automatic guided vehicles with an inclined steering column and an offset distance: criterion for existence of inverse kinematic solution. *Journal of Robotic Systems*, 9:1059–81, December 1992.

[23] H.-D. Chiang, J.-C. Wang, and O. Cockings. Optimal capacitor placements in distribution systems. *IEEE Transactions on Power Delivery*, 5:634–49, April 1990.

[24] Y. J. Choi, C. D., 3rd. Crane, and G. K. Matthew. Obstacle avoidance via articulation. *Journal of Robotic Systems*, 8:465–84, August 1991.

[25] K. F. Cook and R. J. Cipra. An obstacle detection and avoidance algorithm for a planar manipulator. *ASME Advances in Design Automation*, DE 19-1:353–360, 1989.

[26] I. J. Cox. Blanche–an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7:193–204, April 1991.

[27] M. Cox. The numerical evaluation of b-splines. *J. Inst. Maths. Applic.*, 10:134–149, 1972.

[28] C. W. de Silva. Trajectory design for robotic manipulators in space applications. *Journal of Guidance, Control, and Dynamics*, 14:670–4, May/June 1991.

[29] C. deBoor. *A practical guide to splines.* Springer Verlag, New York, 1978.

[30] J.E. Dennis and R.B. Schnable. *Numerical Methods For Unconstrained Optimization And Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, N.J., 1983.

[31] S. Devadas and A. R. Newton. Genie: A generalized array optimizer for vlsi synthesis. In *Proc. 23rd Decision and Automatin Conf.*, pages 631–637, Las Vegas, June 1986.

[32] S. Devadas and A. R. Newton. Topological optimization of multiple-level array logic. *IEEE Trans. on Computer-Aided Design*, CAD-6(6):915–941, 1987.

[33] W. B. Dolan, P. T. Cummings, and M. D. LeVan. Process optimization via simulated annealing: application to network design. *AIChE Journal*, 35:725–36, May 1989.

[34] B.R. Donald. A search algorithm for motion planning. *Artificial Intelligence*, 31:295–353, 1987.

[35] G. Dudek, M. Jenkin, and E. Milios. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7:859–65, December 1991.

[36] R. E. Ellis. Geometric uncertainties in polyhedral object recognition. *IEEE Transactions on Robotics and Automation*, 7:361–71, June 1991.

[37] R. E. Ellis. Planning tactile recognition paths in two and three dimensions. *International Journal of Robotics Research*, 11:87–111, April 1992.

[38] T. Elperin. Monte carlo structural optimization in discrete variables with annealing algorithm. *Int. J. for Numerical Methods in Eng.*, 26:815–821, 1988.

[39] U. Faigle and W. Kern. Note on the convergence of simulated annealing algorithms. *SIAM Journal on Control and Optimization*, 29:153–9, January 1991.

[40] H. Fleisher, D.B. Giraldi, R.L. Phoenix, and M.A. Tavel. Simulated annealing as a tool for logic optimization in a cad environment. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, pages 203–205, Santa Clara, November 1985.

[41] G. N. Frederickson and D. J. Guan. Preemptive ensemble motion planning on a tree. *SIAM Journal on Computing*, 21:1130–52, December 1992.

[42] T. L. Friesz, H.-J. Cho, and N. J. Mehta. A simulated annealing approach to the network design problem with variational inequality constraints. *Transportation Science*, 26:18–26, February 1992.

[43] M. Galicki. Optimal planning of a collision-free trajectory of redundant manipulators. *International Journal of Robotics Research*, 11:549–59, December 1992.

[44] S. B. Gelfand and S. K. Mitter. Analysis of simulated annealing for optimization. In *Proc. 24th Conf. on Decision and Control*, pages 779–786, Ft. Lauderdale, December 1985.

[45] G. G. E. Gielen, Herman C. C. Walscharts, and W. M. C. Sansen. Analog circuit design optimization based on symbolic simulation and simulated annealing. *IEEE Journal of Solid-State Circuits*, 25:707– 13, June 1990.

[46] E. G. Gilbert and D.W. Johnson. Distance functions and their application to robot path planning in the presence of obstacles. *IEEE J. of Robotics and Automation*, RA-1(1):20–30, 1985.

[47] B.L. Golden and C.C. Skiscim. Using simulated annealing to solve routing and location problems. *Naval Logistics Research Quarterly*, 33:261–279, 1986.

[48] G. Gonsalves. Logic synthesis using simulated annealing. In *Proc. IEEE Int. Conf. on Computer Design*, pages 561–564, Port Chester, October 1986.

[49] J.W. Greene and K.J. Supowit. Simulated annealing without rejected moves. *IEEE Trans. on Computer-Aided Design*, CAD-5:221–228, 1986.

[50] L.K. Grover. A new simulated annealing algorithm for standard cell placement. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, pages 378–380, Santa Clara, November 1986.

[51] Z. Y. Guo and T. C. Hsia. Joint trajectory generation for redundant robots in an environment with obstacles. *Journal of Robotic Systems*, 10:199–215, March 1993.

[52] D. Halperin, M. H. Overmars, and M. Sharir. Efficient motion planning for an l-shaped object. *SIAM Journal on Computing*, 21:1–23, February 1992.

[53] T. Hamada, C.-K. Cheng, and P. M. Chau. An efficient multilevel placement technique using hierarchical partitioning. *IEEE Transactions on Circuits and Systems. Part I, Fundamental Theory and Applications*, 39:432–9, June 1992.

[54] H. Haneishi, T. Masuda, and N. Ohyama. Analysis of the cost function used in simulated annealing for ct image reconstruction. *Applied Optics*, 29:259–65, January 1990.

[55] C. W. Hasselfield, P. Wilson, and L. Penner. An automated method for least cost distribution planning. *IEEE Transactions on Power Delivery*, 5:1188–94, April 1990.

[56] T. Hongo and H. Arakawa. An automatic guidance system of a self-controlled vehicle... In *Proc. IECON*, 1985.

[57] M.D. Huang, F. Romeo, and A.L. Sangiovanni-Vincentelli. An efficient general cooling schedule for simulated annealing. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, pages 381–384, Santa Clara, November 1986.

[58] Y. K. Hwang and N. Ahuja. Gross motion planning–a survey. *Computing Surveys*, 24:219–91, September 1992.

[59] Y. K. Hwang and N. Ahuja. A potential field approach to path planning. *IEEE Transactions on Robotics and Automation*, 8:23–32, February 1992.

[60] M. R. Irving and M. J. H. Sterling. Optimal network tearing using simulated annealing. *IEE Proceedings. Part C, Generation, Transmission and Distribution*, 137:69–72, January 1990.

[61] S. D. Jackson and R. O. Mittal. Automatic generation of 2-axis laser-cutter nc machine program and path planning from cad. *Computers in Industry*, 21:223–31, February 1993.

[62] T. Jia and F. M. L. Amirouche. Natural dynamics of robot manipulators in the presence of obstacles. *Journal of Dynamic Systems, Measurement and Control*, 113:170–4, March 1991.

[63] P. Jocobs and J. Canny. Planning smooth paths for mobile robots. *IEEE J. of Robotics and Automation*, pages 2–7, August 1989.

[64] S. Jun and K. G Shin. Shortest path planning in discretized workspaces using dominance relation. *IEEE Transactions on Robotics and Automation*, 7:342–50, June 1991.

[65] M. Kameyama, T. Amada, and T. Higuchi. Highly parallel collision detection processor for intelligent robots. *IEEE Journal of Solid-State Circuits*, 27:500–6, April 1992.

[66] Y. Kanayama and B. Hartman. Smooth local path planning for autonomous vehicles. Technical Report TRCS88-15, University of California, Santa Barbara, 1986.

[67] Y. Kanayama and N. Miyake. Trajectory generation for mobile robots. In *Proc. 3rd Int. Symp. on Robotics Research*, pages 333–340, Gouvieux, France, 1985.

[68] Y. Kanayama and S. Yuta. Vehicle path specification by a sequence of straight lines. *IEEE J. of Robotics and Automation*, 4(3):265–275, June 1988.

[69] K. Kant and S. W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *Int. J. Rob. Res.*, 5(3):72–89, Fall 1986.

[70] A. Khachaturyan. Statistical mechanics approach in mimimizing a multivariable function. *J. Math. Phys.*, 27:1834–1838, 1986.

[71] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. of Robotics Research*, 5(1):90–98, 1986.

[72] J. Khouri and K. A. Stelson. An efficient algorithm for shortest path in three dimensions with polyhedral obstacles. *Journal of Dynamic Systems, Measurement and Control*, 111:433–6, September 1989.

[73] B. K. Kim and K. G. Shin. Minimum-time path planning for robot arms and their dynamics. *IEEE Trans. on Sys., Man and Cyb.*, SMC-15(2):213–223, March 1985.

[74] J.-O. Kim and P. K. Khosla. Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, 8:338–49, June 1992.

[75] M. S. Kim and C. C. Guest. Simulated annealing algorithm for binary phase only filters in pattern classification. *Applied Optics*, 29:1203–8, March 1990.

[76] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[77] S. Kirkpatrick and G. Toulouse. Configuration space analysis of travelling salesman problems. *J. Physique*, 46:1277–1292, 1985.

[78] R. M. Kling and P. Banerjee. Empirical and theoretical studies of the simulated evolution method applied to standard cell placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10:1303–15, October 1991.

[79] Stephen G. Kochan. *Programming in C.* Hayden Books, revised edition, 1992.

[80] K. Komoriya and S. Tachi. A method for autonomous locomotion of mobile robots. *J. of Robotics Soc. Japan*, 2:222–231, 1984.

[81] K. Kondo. Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration. *IEEE Transactions on Robotics and Automation*, 7:267–77, June 1991.

[82] S.A. Kravitz and R. Rutenbar. Multiprocessor-based placement by simulated annealing. In *Proc. 23rd Des. Automation Conf.*, pages 567–573, Las Vegas, June 1986.

[83] D. J. Kropaczek and P. J. Turinsky. In-core nuclear fuel management optimization for pressurized water reactors utilizing simulated annealing. *Nuclear Technology*, 95:9–32, July 1991.

[84] H.-M. Ku and I. Karimi. An evaluation of simulated annealing for batch process scheduling. *Industrial & Engineering Chemistry Research*, 30:163–9, January 1991.

[85] W. A. Kuperman, M. D. Collins, and J. S. Perkins. Optimal time-domain beamforming with simulated annealing including application of a priori information. *The Journal of the Acoustical Society of America*, 88:1802–10, October 1990.

[86] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications.* D. Reidel Pub. Co., Dordrecht, Holland, 1987.

[87] J. Lam and J.-M. Delosme. Optimal annealing schedule. Technical Report 8608, Yale University, New Haven, 1986.

[88] P. Lancaster and K. Salkauskas. *Curve and Surface Fitting.* Academic Press, San Diego, 1986.

[89] J.-C. Latombe, A. Lazanas, and S. Shekhar. Robot motion planning with uncertainty in control and sensing. *Artificial Intelligence*, 52:1–47, November 1991.

[90] David Lawrence. *Genetic algorithms and simulated annealing.* Pitman Publishers; Morgan Kaufmann Publishers, London, Los Altos, Calif., 1987.

[91] E. T. Y. Lee. A simplified b-spline computation routine. *Computing*, 29:365–371, 1983.

[92] T. S. Levitt and D. T. Lawton. Qualitative navigation for mobile robots. *Artificial Intelligence*, 44:305–60, August 1990.

[93] Frank L. Lewis. *Optimal Control.* John Wiley & Sons, Inc., 1986.

[94] T. Lianos, D. Kiritsis, and N. Aspragathos. A direct algorithm for continuous path control of manipulators. *Robotics and Computer-Integrated Manufacturing*, 8(2):97–101, 1991.

[95] C.Y. Liu and R.W. Mayne. Distance calculation in motion planning problems with interference situations. *ASME Advances in Design Automation*, DE-Vol. 23-1:145–152, 1990.

[96] Y.-H. Liu and S. Arimoto. Path planning using a tangent graph for mobile robots among polygonal and curved obstacles. *International Journal of Robotics Research*, 11:376–82, August 1992.

[97] T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Trans. on Computers*, C-32(2):108–120, 1983.

[98] T. Lozano-Perez. A simple motion planning algorithm for general robotic manipulators. *IEEE J. of Robotics and Automation*, RA-3(3), June 1987.

[99] T. Lozano-Perez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[100] J. Y. S. Luh and C. E. Campbell. Minimum distance collision-free path planning for industrial robots with a prismatic joint. *IEEE Trans. on Automatic Control*, 29(8):675–680, August 1984.

[101] J. Y. S. Luh and C. S. Lin. Optimum path planning for mechanical manipulators. *J. Dyn. Sys., Meas., and Ctrl.*, 102:142–151, June 1981.

[102] V. Lumelsky and K. Sun. A unified methodology for motion planning with uncertainty for 2d and 3d two-link robot arm manipulators. *Int. J. of Rob. Res.*, 9(5):89–104, October 1990.

[103] V. J. Lumelsky. A comparative study on the path length performance of maze- searching and robot motion planning algorithms. *IEEE Transactions on Robotics and Automation*, 7:57–66, February 1991.

[104] M. Lundy and A. Mees. Convergence of an annealing algorithm. *Math. Prog.*, 34:111–124, 1986.

[105] A. Malhotra, J. H. Oliver, and W. Tu. Synthesis of spatially and intrinsically constrained curves using simulated annealing. *ASME Advances in Design Automation*, DE-32:145–155, 1991.

[106] N. Metropolis, A. Rosenbluth, M. Rosenbluth, and A. Teller. Equations of state calculations by fast computing machines. *J. of Chemical Physics*, 21:1087–1091, 1953.

[107] J. del R. Millan and C. Torras. A reinforcement connectionist approach to robot path finding in non-maze-like environments. *Machine Learning*, 8:363–95, May 1992.

[108] J. S. B. Mitchell and C. H. Papadimitriou. The weighted region problem: finding shortest paths through a weighted planar subdivision. *Journal of the Association for Computing Machinery*, 38:18–73, January 1991.

[109] T. P. Moore and A. J. de Geus. Simulated annealing controlled by a rule-based expert system. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, pages 200–202, Santa Clara, November 1985.

[110] M. E. Mortenson. *Geometric Modeling.* Wiley, New York, 1985.

[111] H. Moulin and E. Bayo. On the accuracy of end-point trajectory tracking for flexible arms by noncausal inverse dynamic solutions. *Journal of Dynamic Systems, Measurement and Control*, 113:320–4, June 1991.

[112] Y. Nakamura and R. Mukherjee. Nonholonomic path planning of space robots via a bidirectional approach. *IEEE Transactions on Robotics and Automation*, 7:500–14, August 1991.

[113] W. Newman and M. Branicky. Real-time configuration space transforms for obstacle avoidance. *Int. J. of Rob. Res.*, 10(6):650–667, December 1991.

[114] Parviz E. Nikravesh. *Computer-Aided Analysis of Mechanical Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1988.

[115] M. Okutomi and M. Mori. Decision of robot movement bye means of a potential field. *JRSJ Advanced Robotics*, 1(2):131–141, 1986.

[116] R.H.J.M. Otten and L.P.P.P. van Ginneken. *The annealing algorithm.* Kluwer Academic Publishers, Boston, 1989.

[117] I. Pandelidis and Q. Zou. Optimization of injection molding design. *Polymer Engineering and Science*, 30:873–92, Mid-August 1990.

[118] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, New York, 1982.

[119] G. T. Parks. An intelligent stochastic optimization routine for nuclear fuel cycle design. *Nuclear Technology*, 89:233–46, February 1990.

[120] A. N. Patel, R. S. H. Mah, and I. A. Karimi. Preliminary design of multiproduct noncontinuous plants using simulated annealing. *Computers & Chemical Engineering*, 15:451–69, July 1991.

[121] Carl E. Pearson. *Numerical Methods in Engineering and Science.* Van Nostrand Reinhold Co. Ltd., England, 1986.

[122] A. A. Petrov and I. M. Sirota. Control of a robot manipulator with obstacle avoidance under little information about the environment. In *Proc. 8th IFAC Triennial World Congress*, pages 1903–1908, Kyoto, Japan, 1981.

[123] M. Piccioni. A combined multistart-annealing algorithm for continuous global optimization. *Computers & Mathematics with Applications*, 21(6-7):173–9, 1991.

[124] L. Piegl and W. Tiller. Curve and surface contructions usign rational b-splines. *Computer Aided Design*, 19(9):485–498, 1987.

[125] M. Pincus. A monte carlo method for the approximate solution of certain types of constrained optimization problems. *Oper. Res.*, 18:1225–1228, 1970.

[126] W.H. Press, S.A. Teukolsky, W.T. Vtterling, and B.P. Flannery. *Numerical Recipes in C.* Cambridge University Press, second edition, 1992.

[127] N. Radcliffe and G. Wilson. Natural solutions give their best. *New Scientist*, 126:47–50, April 1990.

[128] R.E. Randelman and G.S. Grest. N-city traveling salesman problem: Optimization by simulated annealing. *J. Statis. Phys.*, 45:885–890, 1986.

[129] N. S. V. Rao, N. Stoltzfus, and S. S. Iyengar. A "retraction" method for learned navigation in unknown terrains for a circular robot. *IEEE Transactions on Robotics and Automation*, 7:699–707, October 1991.

[130] W. E. Reddick. Automated spatial filter optimization using frequency-domain simulated annealing. *Optical Engineering*, 31:82–7, January 1992.

[131] E. Rimon and D.E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8:501–18, October 1992.

[132] F. Robuste, C. F. Daganzo, and R. R.,II Souleyrette. Implementing vehicle routing models. *Transportation Research. Part B, Methodological*, 24B:263–86, August 1990.

[133] E. Y. Rodin, B. K. Ghosh, and F. Golenko. Collision-free path-planning in a dynamic environment: semantic control approach. *Simulation*, 51:196–201, November 1988.

[134] C. Rose. Low mean internodal distance network topologies and simulated annealing. *IEEE Transactions on Communications*, 40:1319– 26, August 1992.

[135] C. S. Ruf. Numerical annealing of low-redundancy linear arrays. *IEEE Transactions on Antennas and Propagation*, 41:85–9, January 1993.

[136] R.A. Rutenbar. Simulated annealing algorithms: An overview. *IEEE Circuits and Devices*, pages 19–26, January 1989.

[137] G. Sahar and J. M. Hollerbach. Planning of minimum-time trajectories for robot arms. *Int. J. Rob. Res.*, 5(3):90–100, Fall 1986.

[138] T. Sakurai, B. Lin, and A. R. Newton. Fast simulated diffusion: an optimization algorithm for multiminimum problems and its applications to mosfet model parameter extraction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11:228–34, February 1992.

[139] C. Samson. Time-varying feedback stabilization of car-like wheeled mobile robots. *International Journal of Robotics Research*, 12:55–64, February 1993.

[140] E. Sandgren and S. Vekataraman. Robot path planning via simulated annealing: A near real-time approach. *ASME Advances in Design Automation*, DE 19-1:345–351, 1989.

[141] T. Satoh and K. Nara. Maintenance scheduling by using simulated annealing method. *IEEE Transactions on Power Systems*, 6:850–6, May 1991.

[142] B. Schmult. Autonomous robotic disassembly in the blocks world. *International Journal of Robotics Research*, 11:437–59, October 1992.

[143] A. Schweikard. Polynomial time collision detection for manipulator paths specified by joint motions. *IEEE Transactions on Robotics and Automation*, 7:865–70, December 1991.

[144] C. A. Shaffer and G.M. Herb. A real-time robot arm collision avoidance system. *IEEE Transactions on Robotics and Automation*, 8:149–60, April 1992.

[145] R. Sharma. Locally efficient path planning in an uncertain, dynamic environment using a probabilistic model. *IEEE Transactions on Robotics and Automation*, 8:105–10, February 1992.

[146] Z. Shiller and S. Dubowsky. On computing the global time-optimal motions of robotic manipulators in the presence of obstacles. *IEEE Transactions on Robotics and Automation*, 7:785–97, December 1991.

[147] Z. Shiller and Y.-R. Gwo. Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 7:241–9, April 1991.

[148] Z. Shiller and H.-H. Lu. Computation of path constrained time optimal motions with dynamic singularities. *Journal of Dynamic Systems, Measurement and Control*, 114:34–40, March 1992.

[149] K. G. Shin and Q. Zheng. Minimum-time collision-free trajectory planning for dual-robot systems. *IEEE Transactions on Robotics and Automation*, 8:641–4, October 1992.

[150] S.K. Singh and M. C. Leu. Manipulator motion planning in the presence of obstacles and dynamic constraints. *Int. J. of Rob. Res.*, 10(2):171–187, April 1991.

[151] R. L. Somorjai. Novel approach for computing the global minimum of proteins. general concepts, methods, and approximations. *The Journal of Physical Chemistry*, 95:4141–6, May 1991.

[152] Mark W. Spong and M. Vidyasagar. *Robot Dynamics and Control.* John Wiley & Sons, Inc., 1989.

[153] K. V. Steiner, M. Keefe, and A. Wolff. Interactive graphics simulation with multi-level collision algorithm. *Journal of Manufacturing Systems*, 11(6):462–9, 1992.

[154] T. Subbian, D. R. Flugrad, and P. Kavanagh. Robot trajectory planning by a continuation method. *Factory Auto. and Info. Management*, 1990.

[155] S.-H. Suh, I.-K. Woo, and S.-K. Noh. Automatic trajectory planning system (atps) for spray painting robots. *Journal of Manufacturing Systems*, 10(5):396–406, 1991.

[156] K. Sun and V. Lumelsky. Path planning among unknown obstacles: the case of a three-dimensional cartesian arm. *IEEE Transactions on Robotics and Automation*, 8:776–86, December 1992.

[157] S. Sutanthavibul, E. Shragowitz, and J. B. Rosen. An analytical approach to floorplan design and optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10:761–9, June 1991.

[158] J. M. Sutter and J. H. Kalivas. Convergence of generalized simulated annealing with variable step size with application toward parameter estimations of linear and nonlinear models. *Analytical Chemistry*, 63:2383–6, October 1991.

[159] S. Swanson and J. Garner. Applications of newtonian mechanics to curve fitting. *American Journal of Physics*, 57:698–704, August 1989.

[160] M. Szilagyi. Synthesis of electrostatic lenses by simulated annealing. *Journal of Applied Physics*, 66:5087–9, November 1989.

[161] R. Talluri and J. K. Aggarwal. Position estimation for an autonomous mobile robot in an outdoor environment. *IEEE Transactions on Robotics and Automation*, 8:573–84, October 1992.

[162] H. L. Tan, S. B. Gelfand, and E. J. Delp. A cost minimization approach to edge detection using simulated annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:3–18, January 1992.

[163] R. Taylor. Planning and execution of straight line manipulator trajectories. *IBM J. Res. Develop.*, 23(4):424–436, 1979.

[164] Bruce Tidor. Simulated annealing on free energy surfaces by a combined molecular dynamics and monte carlo approach. *The Journal of Physical Chemistry*, 97:1069– 73, February 1993.

[165] T. Tsumura and N. Fujiwara. An experimental system for automatic guidance of robotted vehicle following the route stored in memory. In *Proc. 11th Int. Sym. on Industrial Robots*, pages 187–193, October 1981.

[166] M. M. Van Hulle and G. A. Orban. Representation and processing in a stochastic neural network: an integrated approach. *Neural Networks*, 4(5):643–55, 1991.

[167] .D. Vanderbilt and S.G. Louie. A monte carlo simulated annealing approach to optmization over continuous variables. *J. Comput. Phys.*, 36:259–271, 1984.

[168] K. Vasudevan, W. G. Wilson, and W. G. Laidlaw. Simulated annealing statics computation using an order-based energy function. *Geophysics*, 56:1831–9, November 1991.

[169] Y. V. Venkatesh. Hermite polynomials for signal reconstruction from zero-crossings. *IEE Proceedings. Part I, Communications, Speech and Vision*, 139:587–96, December 1992.

[170] K. Venkatesh Prasad, R. J Mammone, and J. Yogeshwar. Three-dimensional image restoration using constrained optimization techniques. *Optical Engineering*, 29:279–88, April 1990.

[171] S. D. Voliotis, G. I. Panopoulos, and M. A. Christodoulou. Co-ordinated path tracking by two robot arms with a noninverting algorithm based on the simplex method. *IEE Proceedings. Part D, Control Theory and Applications*, 137:390–6, November 1990.

[172] D. Wang and Y. Hamam. Optimal trajectory planning of manipulators with collision detection and avoidance. *International Journal of Robotics Research*, 11:460–8, October 1992.

[173] L.-L. Wang and W.-H. Tsai. Collision avoidance by a modified least-mean-square-error classification scheme for indoor autonomous land vehicle navigation. *Journal of Robotic Systems*, 8:677–98, October 1991.

[174] C. W. Warren, J. C. Danos, and B. W. Mooring. An approach to manipulator path planning. *Int. J. of Rob. Res.*, pages 87–95, 1989.

[175] C.W. Warren. Global path planning using artificial potential field. *IEEE Journal of Robotics and Automation*, pages 316–321, 1989.

[176] C.W. Warren. A technique for autonomous underwater vehicle route planning. *IEEE Journal of Oceanic Engineering*, 15:199– 204, July 1990.

[177] J. L. Wayman. Optimization of signal distribution networks using simulated annealing. *IEEE Transactions on Communications*, 40:465–71, March 1992.

[178] C. R. Weisbin, G. de Saussure, and J. R. Einstein. Autonomous mobile robot navigation and learning. *Computer*, 22:29–35, June 1989.

[179] G. J. Wiens and M. J. Berggren. Suboptimal path planning of robots: minimal nonlinear forces and energy. *Journal of Dynamic Systems, Measurement and Control*, 113:748–52, December 1991.

[180] S. S. Wilson. Teaching network connectivity using simulated annealing on a massively parallel processor. *Proceedings of the IEEE*, 79:559–66, April 1991.

[181] E. E. Witte, R. D. Chamberlain, and M. A. Franklin. Parallel simulated annealing using speculative computation. *IEEE Transactions on Parallel and Distributed Systems*, 2:483–94, October 1991.

[182] D. F. Wong, H.W. Leong, and C.L. Liu. *Simulated annealing for VLSI design.* Kluwer Academic Publishers, Boston, 1988.

[183] T. C. Woo and D. Dutta. Automatic disassembly and total ordering in three dimensions. *Journal of Engineering for Industry*, 113:207–13, May 1991.

[184] C.-H. Wu and C.-C. Jou. Design of a controlled spatial curve trajectory for robot manipulations. *Journal of Dynamic Systems, Measurement and Control*, 113:248–58, June 1991.

[185] R. M. Wygant and B.E. White. Model for robot motion performance. *Journal of Robotic Systems*, 10:141–52, February 1993.

[186] D. C. H. Yang. Collision-free path planning by using non-periodic b-spline curves. *ASME Advances in Design Automation*, pages 201–207, September 1991.

[187] K. Zeger, J. Vaisey, and A. Gersho. Globally optimal vector quantizer design by stochastic relaxation. *IEEE Transactions on Signal Processing*, 40:310–22, February 1992.

[188] H. Zghal, R. V. Dubey, and J. A. Euler. Collision avoidance of a multiple degree of redundancy manipulator operating through a window. *Journal of Dynamic Systems, Measurement and Control*, 114:717–21, December 1992.

[189] C.-D. Zhang and S.-M. Song. Forward kinematics of walking machines with pantograph legs based on selections of independent joints. *Journal of Robotic Systems*, 9:955–71, October 1992.

[190] D. Zhu and J.-C. Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, 7:9–20, February 1991.

[191] Q. Zhu. Hidden markov model for dynamic obstacle avoidance of mobile robot navigation. *IEEE Transactions on Robotics and Automation*, 7:390–7, June 1991.

[192] F. Zhuang and F. D. Galiana. Unit commitment by simulated annealing. *IEEE Transactions on Power Systems*, 5:311–18, February 1990.

APPENDIX A. OUTPUT FOR FIGURES 2.4-2.12

A.1 Output for Figure 2.4

A.1.1 Using Area as Objective Function

```
Optimal fitting of an ellipse to an object in 2D
================================================

Varbl State      Value      Lower Bound    Upper Bound    Lagr Mult   Residual
V  1    FR     1.41421       0.000000E+00      None       0.0000E+00    1.414
V  2    FR    0.707107       0.000000E+00      None       0.0000E+00  0.7071
V  3    FR     1.00000          None           None       0.0000E+00  1.0000E+15
V  4    FR    0.500000          None           None       0.0000E+00  1.0000E+15
V  5    FR     1.00000        -1.00000        1.00000     0.0000E+00  0.0000E+00
V  6    FR  1.793372E-17      -1.00000        1.00000     0.0000E+00    1.000

Final nonlinear objective value =     1.141593

Theta =       0.0000000000000 rad =       0.0000000000000 deg
dx    =       1.0000000000000
dy    =       0.5000000000000
```

A.1.2 Using Error as Objective Function

```
Optimal fitting of a 2D ellipsoid to an object
==============================================

Varbl State      Value      Lower Bound    Upper Bound    Lagr Mult   Residual
V  1    FR     1.26278       0.000000E+00      None       0.0000E+00    1.263
V  2    FR    0.818803       0.000000E+00      None       0.0000E+00  0.8188
V  3    FR    1.000000          None           None       0.0000E+00  1.0000E+15
V  4    FR    0.500000          None           None       0.0000E+00  1.0000E+15
V  5    FR     1.00000        -1.00000        1.00000     0.0000E+00  0.0000E+00
```

```
    V  6    FR  -1.335699E-16  -1.00000       1.00000       0.0000E+00  1.0000

10  Exit E04UCF - Optimal solution found.

    Final nonlinear objective value =    29.54187

    Theta =       0.0000000000000 rad =       0.0000000000000 deg
    dx    =       1.0000000000000
    dy    =       0.5000000000000
```

A.2 Output for Figure 2.5

A.2.1 Using Area as Objective Function

```
    Optimal fitting of an ellipse to an object in 2D
    ================================================

    Varbl State      Value      Lower Bound    Upper Bound    Lagr Mult   Residual
    V  1    FR     1.41421       0.000000E+00      None       0.0000E+00   1.414
5   V  2    FR    0.707107       0.000000E+00      None       0.0000E+00  0.7071
    V  3    FR     1.00000          None           None       0.0000E+00  1.0000E+15
    V  4    FR    0.500000          None           None       0.0000E+00  1.0000E+15
    V  5    FR     1.00000        -1.00000        1.00000      0.0000E+00  0.0000E+00
    V  6    FR   4.316403E-17     -1.00000        1.00000      0.0000E+00   1.000

10  Final nonlinear objective value =    1.141593

    Theta =       0.0000000000000 rad =       0.0000000000000 deg
    dx    =       1.0000000000000
    dy    =       0.5000000000000
```

A.2.2 Using Error as Objective Function

```
    Optimal fitting of a 2D ellipsoid to an object
    ==============================================

    Varbl State      Value      Lower Bound    Upper Bound    Lagr Mult   Residual
    V  1    FR     1.44224       0.000000E+00      None       0.0000E+00   1.442
5   V  2    FR    0.693880       0.000000E+00      None       0.0000E+00  0.6939
    V  3    FR     1.00000          None           None       0.0000E+00  1.0000E+15
    V  4    FR    0.500000          None           None       0.0000E+00  1.0000E+15
    V  5    FR     1.00000        -1.00000        1.00000      0.0000E+00  0.0000E+00
    V  6    FR  -4.781016E-10     -1.00000        1.00000      0.0000E+00  1.0000
```

```
 Exit E04UCF - Optimal solution found.

 Final nonlinear objective value =    0.3730512E-15

 Theta =     -0.0000000004781 rad =      -0.0000000273932 deg
 dx    =      1.0000000032303
 dy    =      0.4999999985046
```

A.3 Output for Figure 2.6

A.3.1 Using Area as Objective Function

```
 Optimal fitting of an ellipse to an object in 2D
 ================================================

   Varbl State      Value      Lower Bound    Upper Bound    Lagr Mult   Residual
   V  1    FR     1.41421       0.000000E+00      None       0.0000E+00    1.414
 5 V  2    FR    0.707105       0.000000E+00      None       0.0000E+00   0.7071
   V  3    FR    0.999996          None           None       0.0000E+00  1.0000E+15
   V  4    FR    0.500000          None           None       0.0000E+00  1.0000E+15
   V  5    FR    0.819153       -1.00000         1.00000     0.0000E+00   0.1808
   V  6    FR    0.573576       -1.00000         1.00000     0.0000E+00   0.4264

10 Final nonlinear objective value =    1.141579

 Theta =      0.6108642782050 rad =      34.9999449964497 deg
 dx    =      0.5323611493116
 dy    =      0.9831494289833
```

A.3.2 Using Error as Objective Function

```
 Optimal fitting of an ellipse to an object
 ==========================================

   Varbl State      Value      Lower Bound    Upper Bound    Lagr Mult   Residual
   V  1    FR     1.26278       0.000000E+00      None       0.0000E+00    1.263
 5 V  2    FR    0.818799       0.000000E+00      None       0.0000E+00   0.8188
   V  3    FR    0.999997          None           None       0.0000E+00  1.0000E+15
   V  4    FR    0.500000          None           None       0.0000E+00  1.0000E+15
   V  5    FR    0.819152       -1.00000         1.00000     0.0000E+00   0.1808
   V  6    FR    0.573576       -1.00000         1.00000     0.0000E+00   0.4264
```

```
 Exit E04UCF - Optimal solution found.

 Final nonlinear objective value =    29.54137

 Theta =      0.6108646475443 rad =     34.9999661580340 deg
 dx    =      0.5323618085604
 dy    =      0.9831502825617
```

A.4 Output for Figure 2.7

A.4.1 Using Area as Objective Function

```
 Optimal fitting of an ellipse to an object in 2D
 ================================================

 Varbl State     Value      Lower Bound    Upper Bound    Lagr Mult   Residual
 V  1    FR    1.41421       0.000000E+00      None       0.0000E+00   1.414
 V  2    FR   0.707105       0.000000E+00      None       0.0000E+00  0.7071
 V  3    FR   0.999996          None           None       0.0000E+00  1.0000E+15
 V  4    FR   0.500000          None           None       0.0000E+00  1.0000E+15
 V  5    FR   0.819153         -1.00000        1.00000    0.0000E+00  0.1808
 V  6    FR   0.573576         -1.00000        1.00000    0.0000E+00  0.4264

 Final nonlinear objective value =    1.141579

 Theta =      0.6108642782049 rad =     34.9999449964484 deg
 dx    =      0.5323611493115
 dy    =      0.9831494289832
```

A.4.2 Using Error as Objective Function

```
 Optimal fitting of an ellipse to an object
 ==========================================

 Varbl State     Value      Lower Bound    Upper Bound    Lagr Mult   Residual
 V  1    FR    1.11803       0.000000E+00      None       0.0000E+00   1.118
 V  2    FR    1.11803       0.000000E+00      None       0.0000E+00   1.118
 V  3    FR   0.533552          None           None       0.0000E+00  1.0000E+15
 V  4    FR   0.982506          None           None       0.0000E+00  1.0000E+15
 V  5    FR   0.999999         -1.00000        1.00000    0.0000E+00  7.3275E-07
 V  6    FR  1.210575E-03      -1.00000        1.00000    0.0000E+00  0.9988

 Exit E04UCF - Optimal solution found.
```

```
Final nonlinear objective value =   0.1581600E-16

Theta =      0.0012105757165 rad =      0.0693608793376 deg
dx    =      0.5323618995304
dy    =      0.9831516781668
```

A.5 Output for Figure 2.8

A.5.1 Using Area as Objective Function

```
Optimal fitting of an ellipse to an object in 2D
================================================

   Varbl State     Value      Lower Bound    Upper Bound    Lagr Mult   Residual
   V  1    FR    2.51150      0.000000E+00      None        0.0000E+00   2.511
5  V  2    FR    1.44813      0.000000E+00      None        0.0000E+00   1.448
   V  3    FR   0.747314         None           None        0.0000E+00  1.0000E+15
   V  4    FR    1.38253         None           None        0.0000E+00  1.0000E+15
   V  5    FR   0.999513       -1.00000         1.00000     0.0000E+00  4.8697E-04
   V  6    FR  -3.120434E-02   -1.00000         1.00000     0.0000E+00  0.9688

10 Final nonlinear objective value =    7.425864

   Theta =     -0.0312094049994 rad =     -1.7881671875782 deg
   dx    =      0.7900912646645
   dy    =      1.3585397653203
```

A.5.2 Using Error as Objective Function

```
Optimal fitting of an ellipse to an object
==========================================

   Varbl State     Value      Lower Bound    Upper Bound    Lagr Mult   Residual
   V  1    FR    2.36397      0.000000E+00      None        0.0000E+00   2.364
5  V  2    FR    1.62287      0.000000E+00      None        0.0000E+00   1.623
   V  3    FR   0.694868         None           None        0.0000E+00  1.0000E+15
   V  4    FR    1.55117         None           None        0.0000E+00  1.0000E+15
   V  5    FR   0.998431       -1.00000         1.00000     0.0000E+00  1.5689E-03
   V  6    FR  -5.599487E-02   -1.00000         1.00000     0.0000E+00  0.9440

10 Exit E04UCF - Optimal solution found.
```

```
Final nonlinear objective value =    1886.890

Theta =     -0.0560241709109 rad =     -3.2099485439165 deg
dx    =      0.7806355865611
dy    =      1.5098306150237
```

A.6 Output for Figure 2.9

A.6.1 Using Area as Objective Function

```
Optimal fitting of an ellipse to an object in 2D
================================================

 Varbl State      Value      Lower Bound    Upper Bound    Lagr Mult   Residual
 V  1    FR     2.51150      0.000000E+00      None       0.0000E+00    2.511
 V  2    FR     1.44813      0.000000E+00      None       0.0000E+00    1.448
 V  3    FR    0.747314         None           None       0.0000E+00  1.0000E+15
 V  4    FR     1.38253         None           None       0.0000E+00  1.0000E+15
 V  5    FR    0.999513       -1.00000        1.00000      0.0000E+00  4.8697E-04
 V  6    FR  -3.120434E-02    -1.00000        1.00000      0.0000E+00  0.9688

Final nonlinear objective value =    7.425864

Theta =     -0.0312094050829 rad =     -1.7881671923615 deg
dx    =      0.7900912646176
dy    =      1.3585397653228
```

A.6.2 Using Error as Objective Function

```
Optimal fitting of an ellipse to an object
==========================================

 V Varbl State      Value      Lower Bound    Upper Bound    Lagr Mult    Residual
 V  1    FR     2.41179      0.000000E+00      None       0.0000E+00    2.412
 V  2    FR     1.55017      0.000000E+00      None       0.0000E+00    1.550
 V  3    FR    0.719976         None           None       0.0000E+00  1.0000E+15
 V  4    FR     1.47949         None           None       0.0000E+00  1.0000E+15
 V  5    FR    0.999024       -1.00000        1.00000      0.0000E+00  9.7558E-04
 V  6    FR  -4.416117E-02    -1.00000        1.00000      0.0000E+00  0.9558

Exit E04UCF - Optimal solution found.

Final nonlinear objective value =    153.6806
```

```
Theta =     -0.0441755346523 rad =     -2.5310716933118 deg
dx    =      0.7846094667805
dy    =      1.4462485316664
```

A.7 Output for Figure 2.10

A.7.1 Using Area as Objective Function

```
Optimal fitting of an ellipse to an object in 2D
================================================

   Varbl State      Value      Lower Bound    Upper Bound    Lagr Mult   Residual
   V  1    FR     2.51149      0.000000E+00      None       0.0000E+00    2.511
5  V  2    FR     1.44813      0.000000E+00      None       0.0000E+00    1.448
   V  3    FR    0.747311         None           None       0.0000E+00  1.0000E+15
   V  4    FR     1.38254         None           None       0.0000E+00  1.0000E+15
   V  5    FR    0.836652      -1.00000         1.00000     0.0000E+00  0.1633
   V  6    FR    0.547734      -1.00000         1.00000     0.0000E+00  0.4523

10 Final nonlinear objective value =    7.425859

   Theta =      0.5796536907209 rad =      33.2117100574879 deg
   dx    =     -0.1320222690121
   dy    =      1.5660294591474
```

A.7.2 Using Error as Objective Function

```
Optimal fitting of an ellipse to an object
==========================================

   Varbl State      Value      Lower Bound    Upper Bound    Lagr Mult   Residual
   V  1    FR     2.36396      0.000000E+00      None       0.0000E+00    2.364
5  V  2    FR     1.62286      0.000000E+00      None       0.0000E+00    1.623
   V  3    FR    0.694863         None           None       0.0000E+00  1.0000E+15
   V  4    FR     1.55117         None           None       0.0000E+00  1.0000E+15
   V  5    FR    0.849986      -1.00000         1.00000     0.0000E+00  0.1500
   V  6    FR    0.526806      -1.00000         1.00000     0.0000E+00  0.4732

10 Exit E04UCF - Optimal solution found.

   Final nonlinear objective value =    1886.883
```

```
Theta =      0.5548382664633 rad =     31.7898909807038 deg
dx    =     -0.2265430441009
dy    =      1.6845328665954
```

A.8 Output for Figure 2.11

A.8.1 Using Area as Objective Function

```
Optimal fitting of an ellipse to an object in 2D
================================================

   Varbl State      Value      Lower Bound    Upper Bound    Lagr Mult   Residual
   V  1    FR     2.51149       0.000000E+00      None       0.0000E+00   2.511
5  V  2    FR     1.44813       0.000000E+00      None       0.0000E+00   1.448
   V  3    FR    0.747311          None           None       0.0000E+00  1.0000E+15
   V  4    FR     1.38254          None           None       0.0000E+00  1.0000E+15
   V  5    FR    0.836652        -1.00000         1.00000    0.0000E+00  0.1633
   V  6    FR    0.547734        -1.00000         1.00000    0.0000E+00  0.4523

10 Final nonlinear objective value =     7.425859

   Theta =      0.5796536909969 rad =     33.2117100733012 deg
   dx    =     -0.1320222693531
   dy    =      1.5660294593774
```

A.8.2 Using Error as Objective Function

```
Optimal fitting of an ellipse to an object
==========================================

   Varbl State      Value      Lower Bound    Upper Bound    Lagr Mult   Residual
   V  1    FR     2.41179       0.000000E+00      None       0.0000E+00   2.412
5  V  2    FR     1.55017       0.000000E+00      None       0.0000E+00   1.550
   V  3    FR    0.719972          None           None       0.0000E+00  1.0000E+15
   V  4    FR     1.47949          None           None       0.0000E+00  1.0000E+15
   V  5    FR    0.843684        -1.00000         1.00000    0.0000E+00  0.1563
   V  6    FR    0.536840        -1.00000         1.00000    0.0000E+00  0.4632

10 Exit E04UCF - Optimal solution found.

   Final nonlinear objective value =     153.6799

   Theta =      0.5666872480012 rad =     32.4687876143518 deg
```

```
dx    =     -0.1868191691595
dy    =      1.6347298636703
```

A.9 Output for Figure 2.12

A.9.1 Using Volume as Objective Function

```
Optimal fitting of an ellipsoid to an object in 3D
==================================================

     Varbl State      Value      Lower Bound    Upper Bound    Lagr Mult   Residual
     V  1    FR     3.07594      0.000000E+00      None       0.0000E+00    3.076
 5   V  2    FR     1.77358      0.000000E+00      None       0.0000E+00    1.774
     V  3    FR    0.692821      0.000000E+00      None       0.0000E+00   0.6928
     V  4    FR    0.747314         None           None       0.0000E+00  1.0000E+15
     V  5    FR     1.38253         None           None       0.0000E+00  1.0000E+15
     V  6    FR    0.400000         None           None       0.0000E+00  1.0000E+15
10   V  7    FR    0.998196       -1.00000         1.00000    0.0000E+00  1.8043E-03
     V  8    FR  -6.004466E-02    -1.00000         1.00000    0.0000E+00   0.9400
     V  9    FR     1.00000       -1.00000         1.00000    0.0000E+00  0.0000E+00
     V 10    FR   1.977743E-18    -1.00000         1.00000    0.0000E+00    1.000
     V 11    FR    0.999583       -1.00000         1.00000    0.0000E+00  4.1675E-04
15   V 12    FR   2.886738E-02    -1.00000         1.00000    0.0000E+00   0.9711

     Final nonlinear objective value =    11.83214

     angX =     -0.0600807965572 rad =     -3.4423760725144deg
     angY =      0.0000000000000 rad =      0.0000000000000deg
     angZ =      0.0288713915593 rad =      1.6542088850157deg
20   d(1) =      0.7900912646696
     d(2) =      1.3585397653230
     d(3) =      0.4000000000000
```